Global Software Distribution with CernVM-FS

Jakob Blomer
CERN

2016 CCL Workshop on Scalable Computing
October 19th, 2016

```
$ cmsRun DiPhoton_Analysis.py
```

The Anatomy of a Scientific Software Stack

(In High Energy Physics)

```
$ cmsRun DiPhoton_Analysis.py
```

My Analysis Code
< 10 Python Classes

CMS Software Framework
$\mathcal{O}(1000)$ C++ Classes

Simulation and I/O Libraries
ROOT, Geant4, MC-*XYZ*

CentOS 6 and Utilities
$\mathcal{O}(10)$ Libraries

```
$ cmsRun DiPhoton_Analysis.py
```

My Analysis Code
< 10 Python Classes

CMS Software Framework
$\mathcal{O}(1000)$ C++ Classes

Simulation and I/O Libraries
ROOT, Geant4, MC-*XYZ*

CentOS 6 and Utilities
$\mathcal{O}(10)$ Libraries

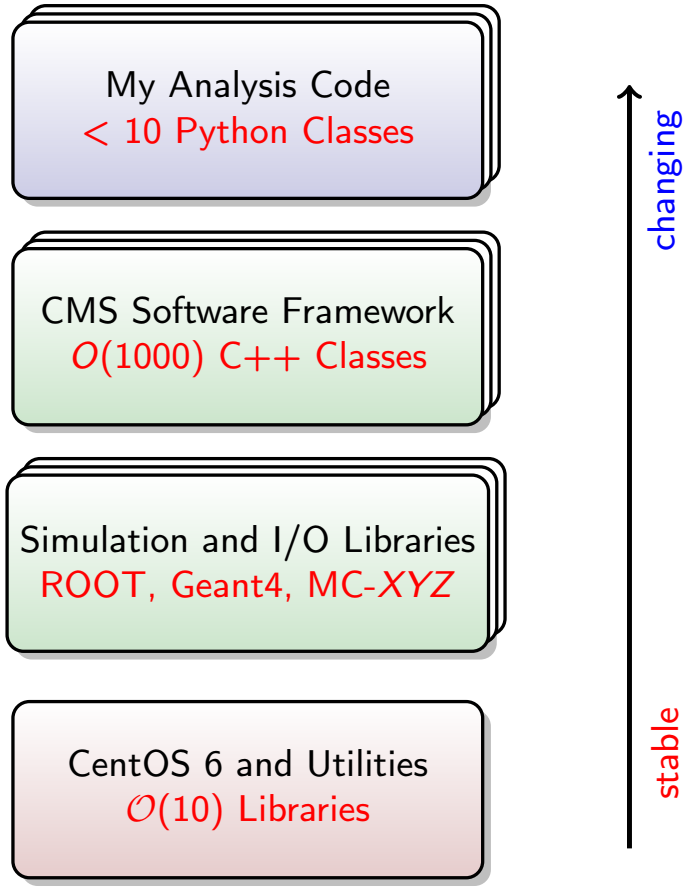
How to install on...

- my laptop:
  compile into `/opt`
  $\sim$ 1 week
- my local cluster:
  ask sys-admin to install in `/nfs/software`
  > 1 week
- someone else's cluster: ?

The Anatomy of a Scientific Software Stack

(In High Energy Physics)

```
$ cmsRun DiPhoton_Analysis.py
```

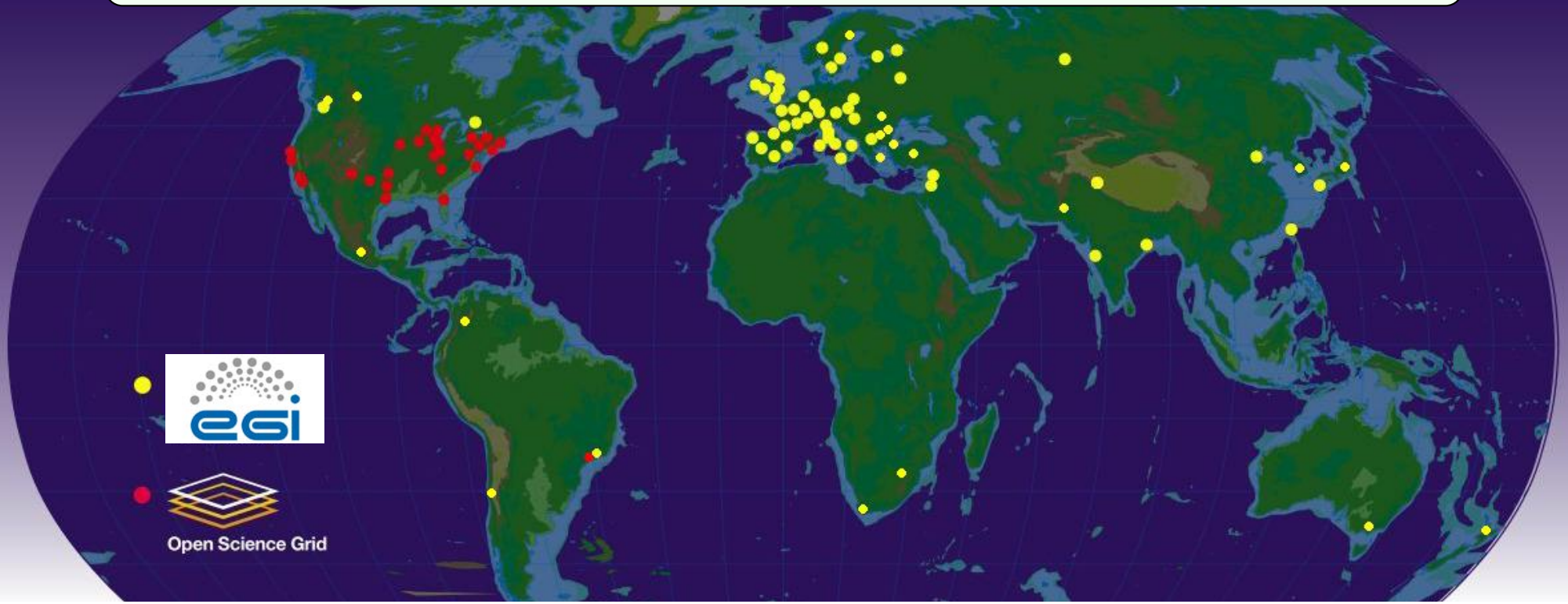


How to install (again) on...

- my laptop:
  compile into `/opt`
  $\sim$ 1 week
- my local cluster:
  ask sys-admin to install in `/nfs/software`
  $>$ 1 week
- someone else's cluster: ?

World Wide LHC Computing Grid

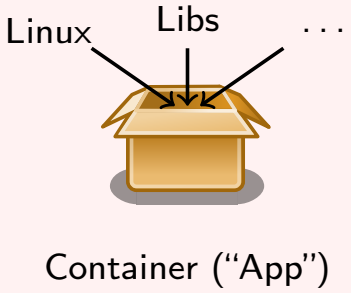
- $\sim$ 200 sites: from 100 to 100 000 cores
- Different countries, institutions, batch schedulers, OSs, ...
- Augmented by clouds, supercomputers, LHC@Home



Example: R in Docker

```
$ docker pull r-base
```
⟶ 1 GB image

```
$ docker run -it r-base
$ ... (fitting tutorial)
```
⟶ only 30 MB used

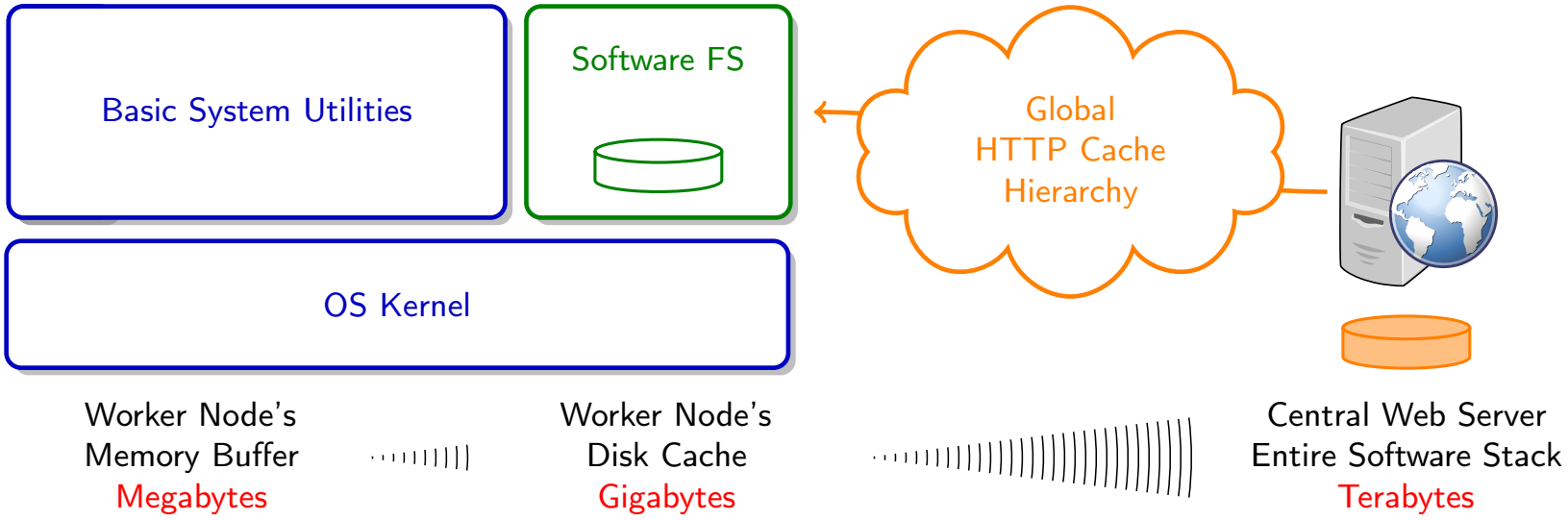


It's hard to scale Docker:

iPhone App	Docker Image
20 MB	1 GB
changes every month	changes twice a week
phones update staggered	servers update synchronized

⟶ Your preferred cluster or supercomputer might not run Docker

A File System for Software Distribution

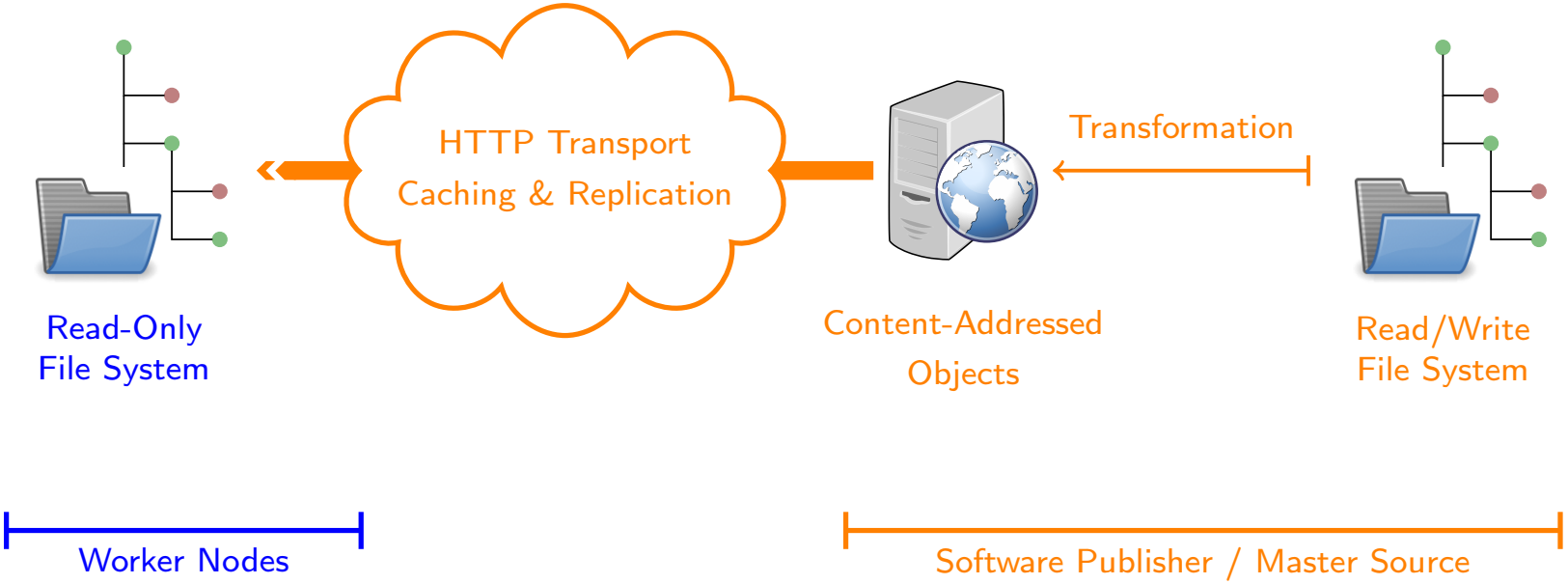


Pioneered by CCL's GROW-FS for CDF at Tevatron

Refined in CernVM-FS, in production for CERN's LHC and other experiments

1. Single point of publishing
2. HTTP transport, access and caching on demand
3. Important for scaling: bulk meta-data download (not shown)

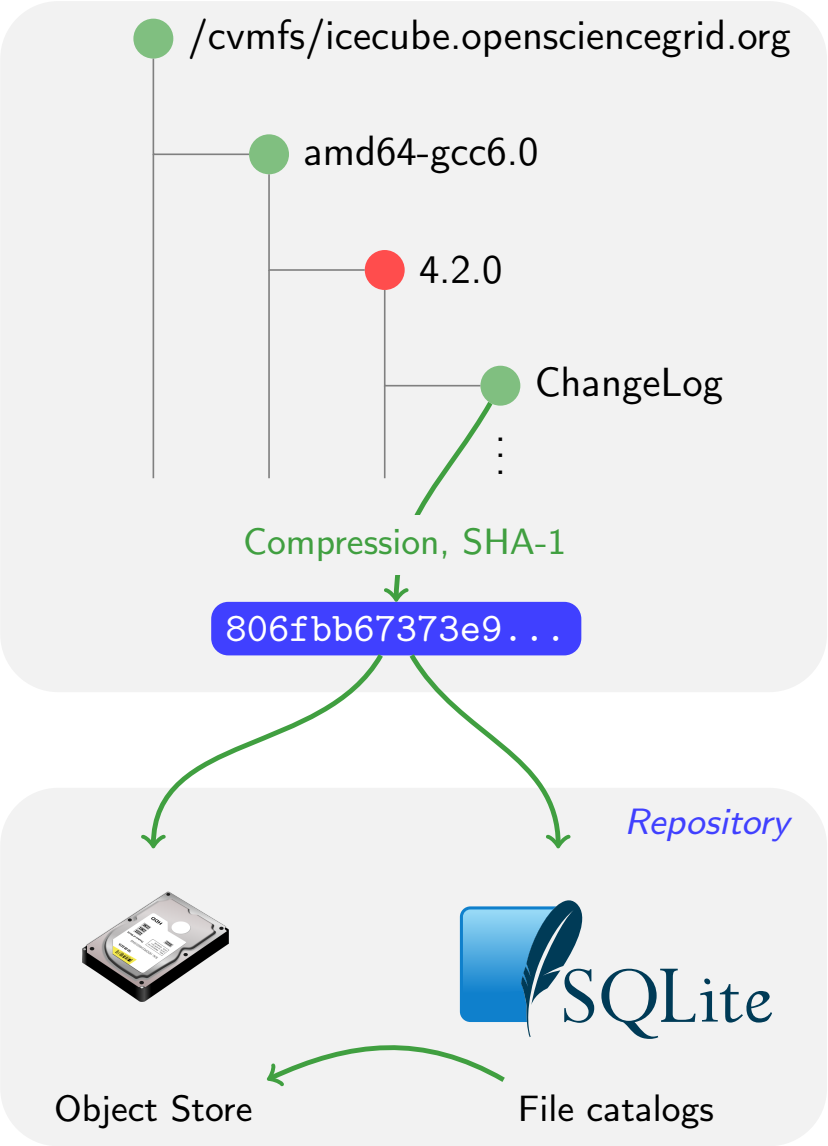
One More Ingredient: Content-Addressable Storage



Two independent issues

1. How to mount a file system (on someone else's computer)?
2. How to distribute immutable, independent objects?

Content-Addressable Storage: Data Structures



Object Store

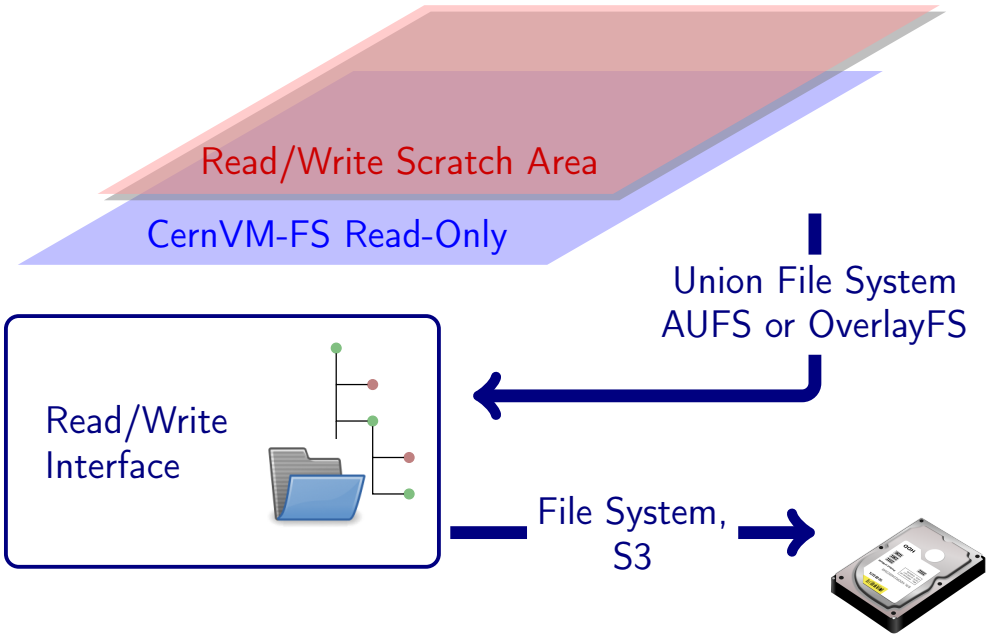
- Compressed files and chunks
- De-duplicated

File Catalog

- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed
  ⇒ integrity, authenticity
- Time to live
- Partitioned / Merkle hashes
  (possibility of sub catalogs)

⇒ **Immutable files, trivial to check for corruption, versioning**

Transactional Publish Interface

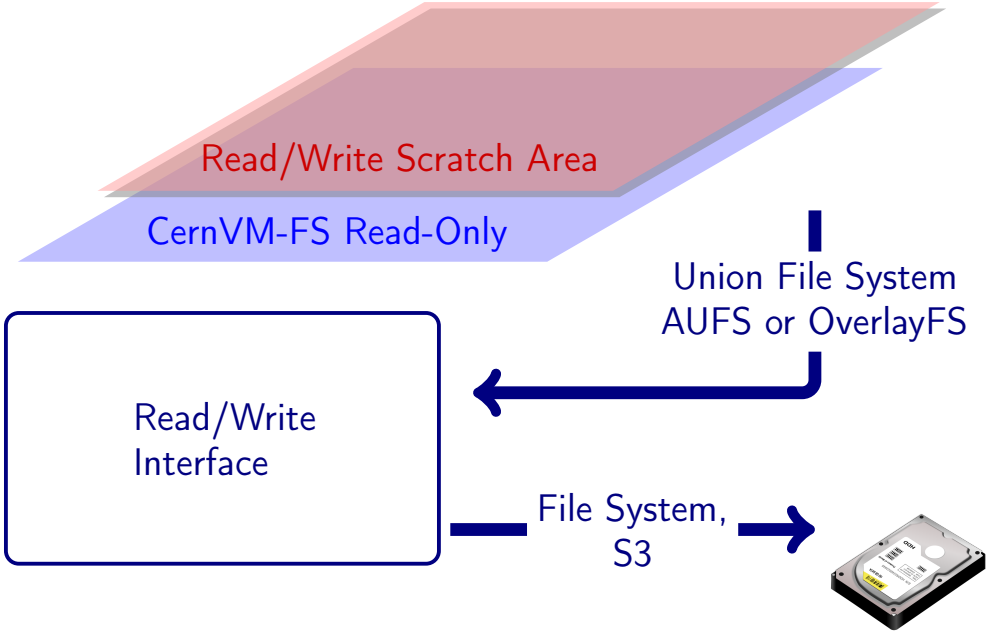


Publishing New Content

```
[ ~ ]# cvmfs_server transaction icecube.opensciencegrid.org
[ ~ ]# make DESTDIR=/cvmfs/opensciencgrid.org/amd64-gcc6.0/4.2.0 install
[ ~ ]# cvmfs_server publish icecube.opensciencegrid.org
```

Uses `cvmfs-server` tools and an Apache web server

Transactional Publish Interface



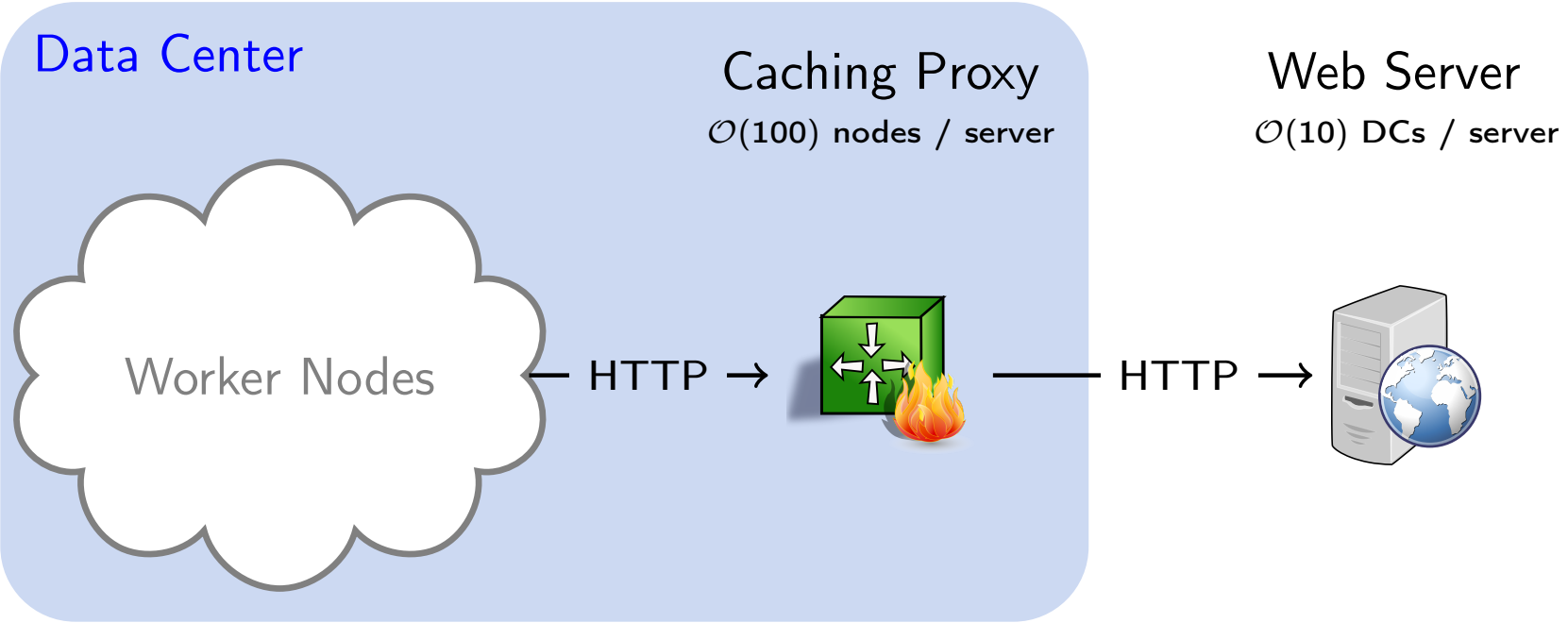
Reproducible: as in git, you can always come back to this state

Publishing New Content

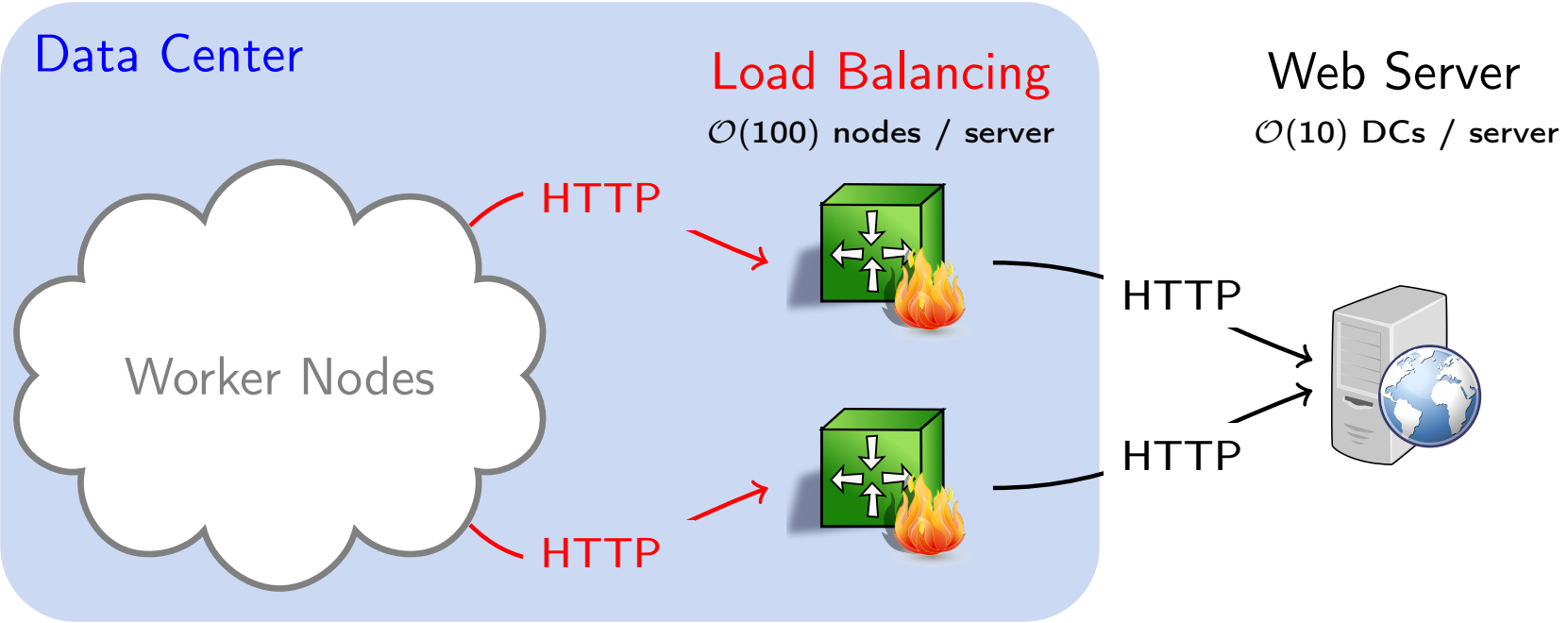
```
[ ~ ]# cvmfs_server transaction icecube.opensciencegrid.org
[ ~ ]# make DESTDIR=/cvmfs/opensciencegrid.org/amd64-gcc6.0/4.2.0 install
[ ~ ]# cvmfs_server publish icecube.opensciencegrid.org
```

Uses `cvmfs-server` tools and an Apache web server

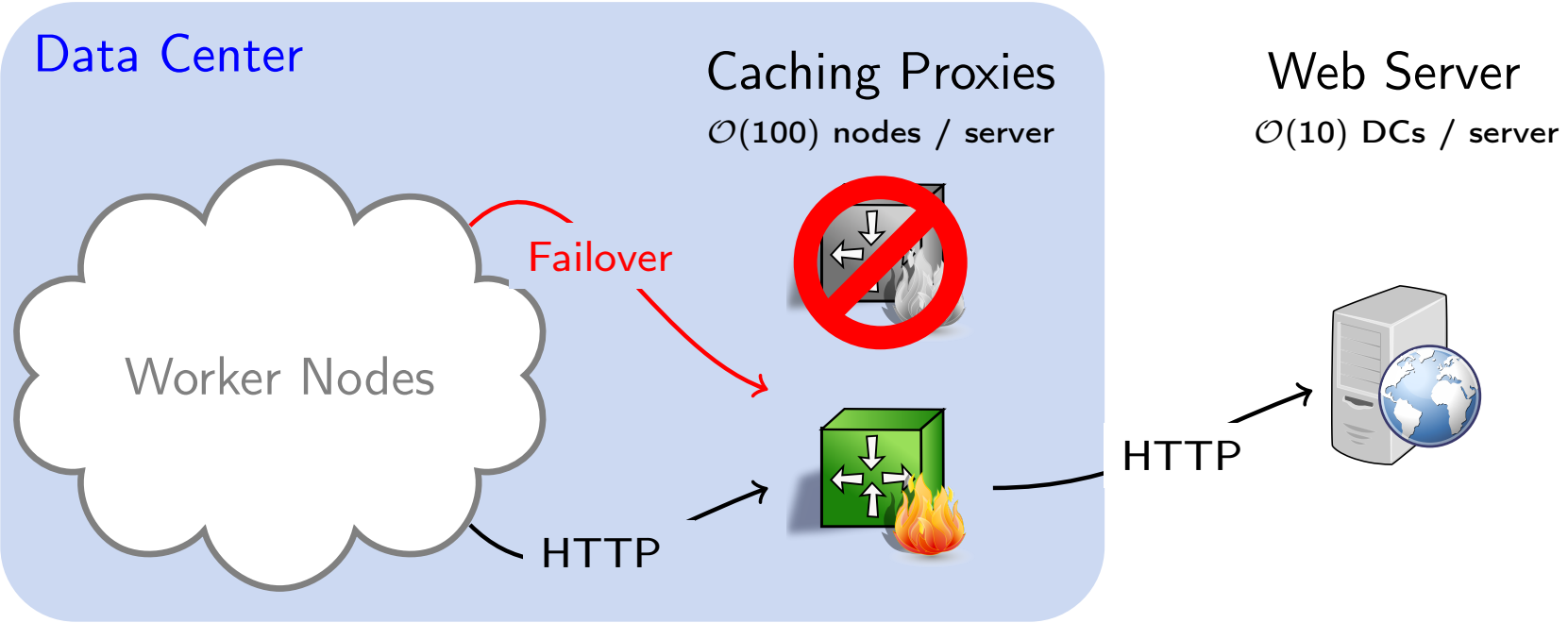
Server side: stateless services



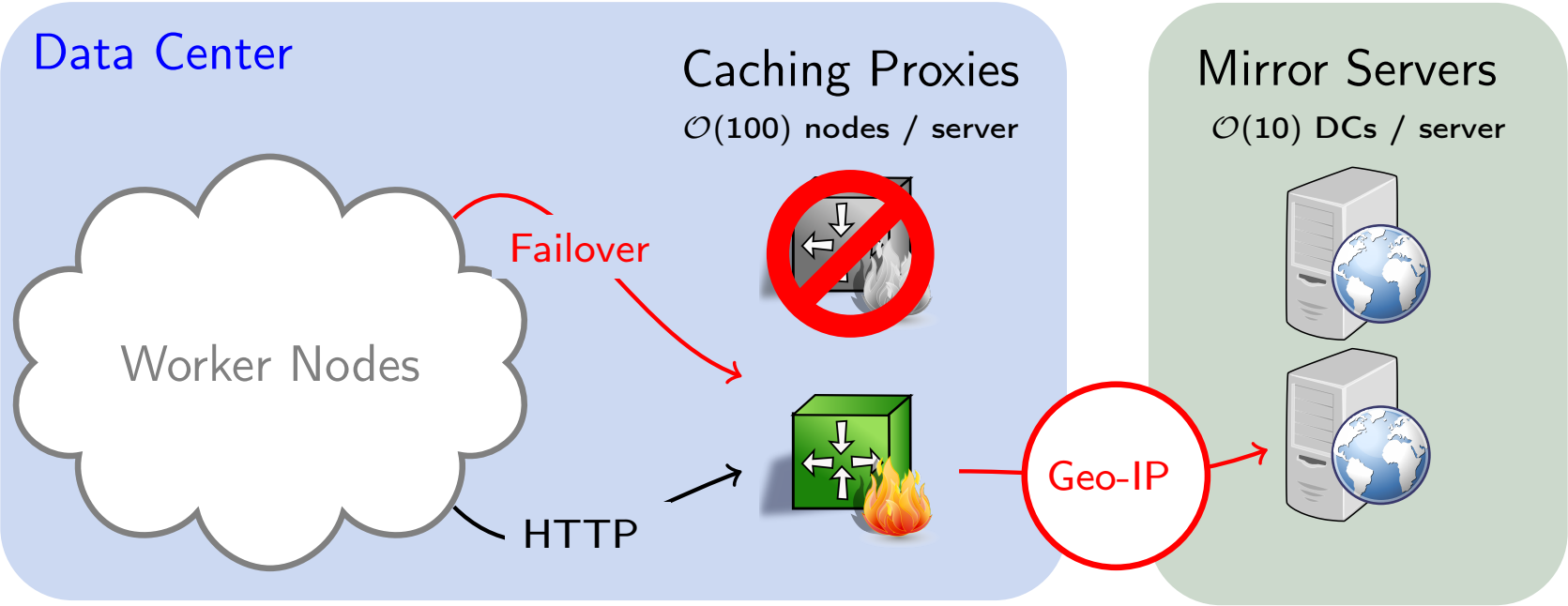
Server side: stateless services



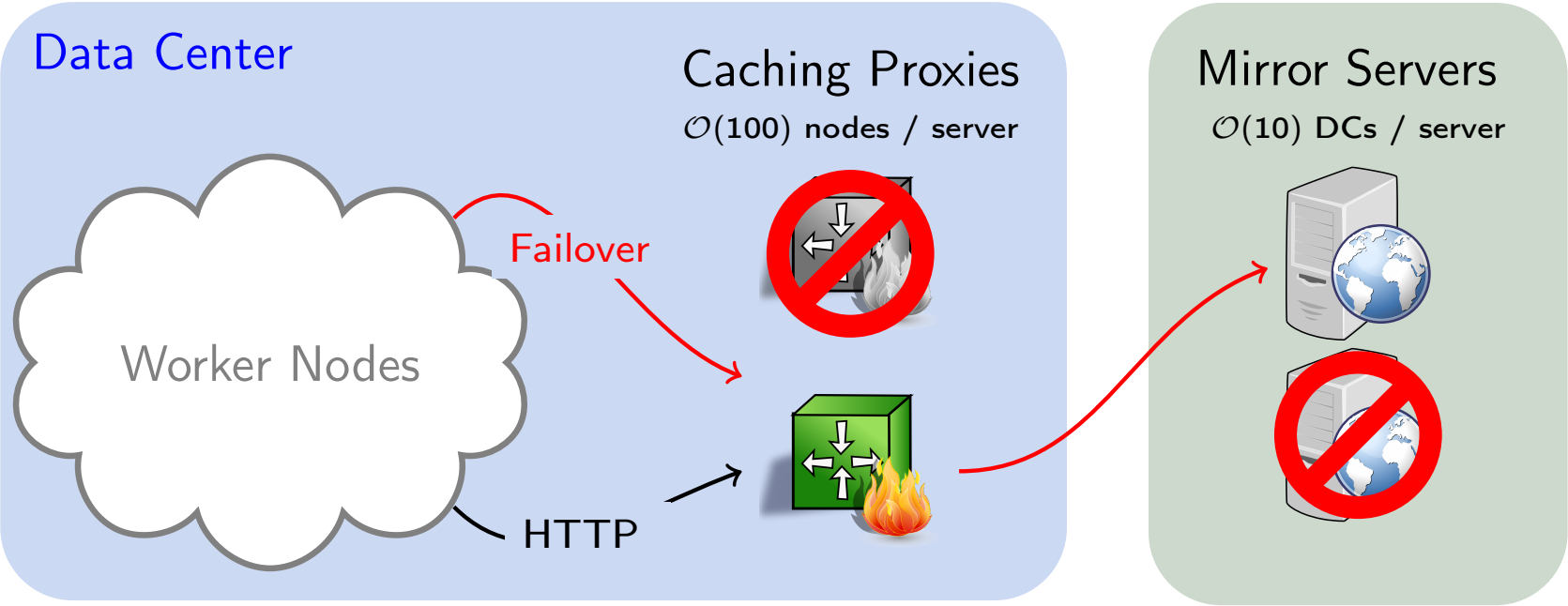
Server side: stateless services



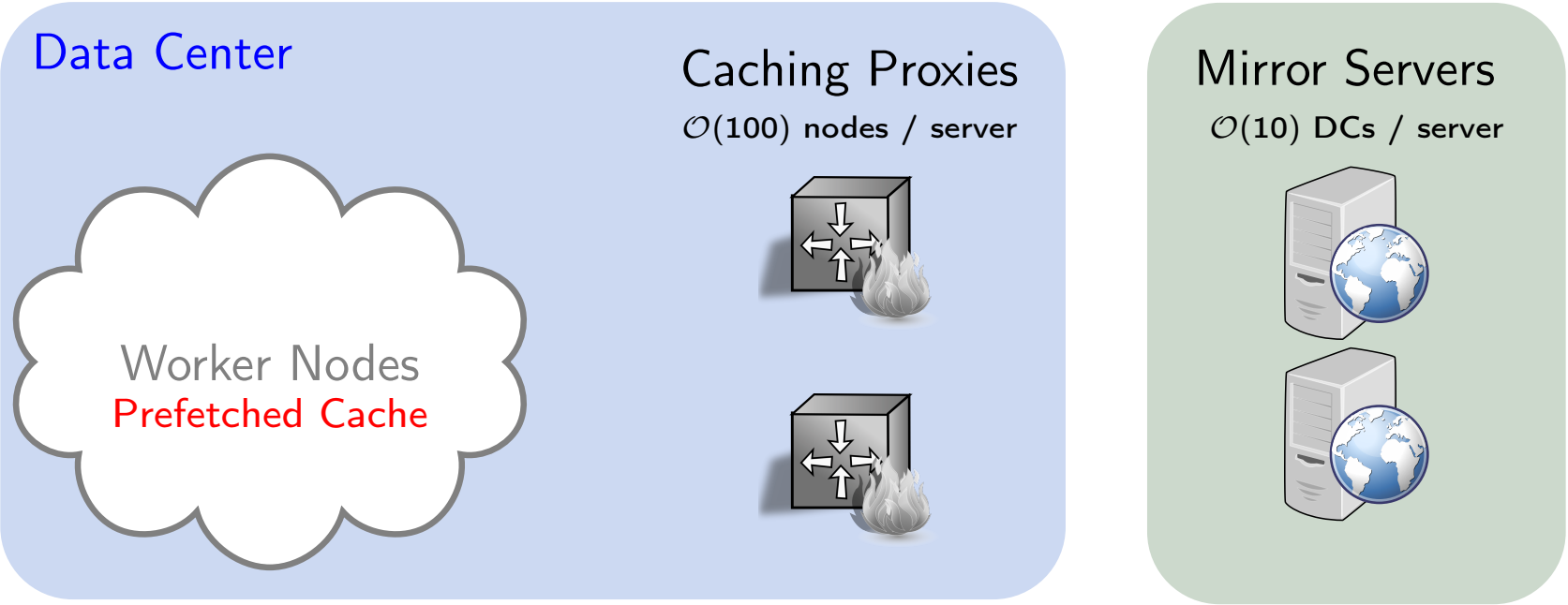
Server side: stateless services



Server side: stateless services



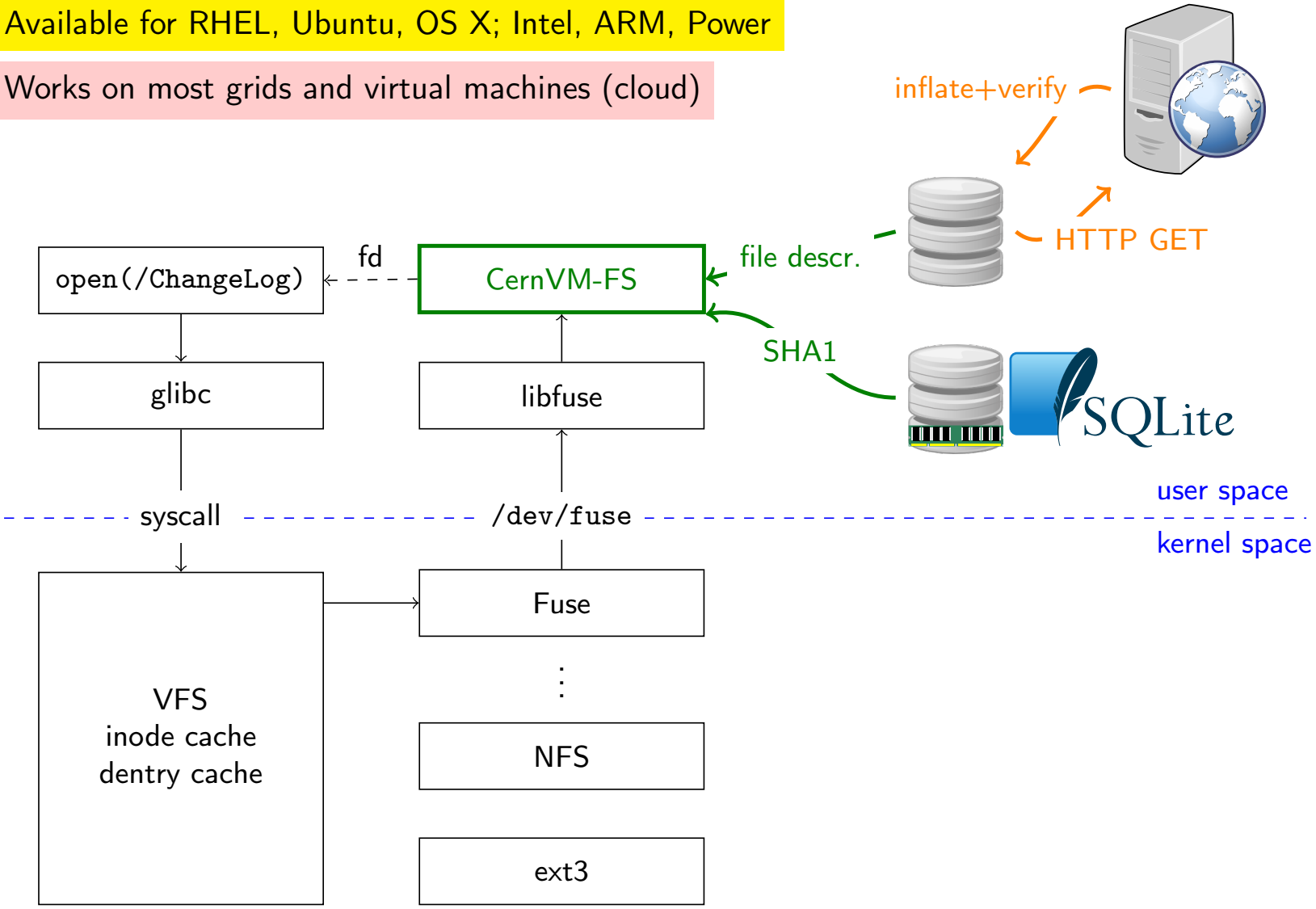
Server side: stateless services



Mounting the File System Client: Fuse

Available for RHEL, Ubuntu, OS X; Intel, ARM, Power

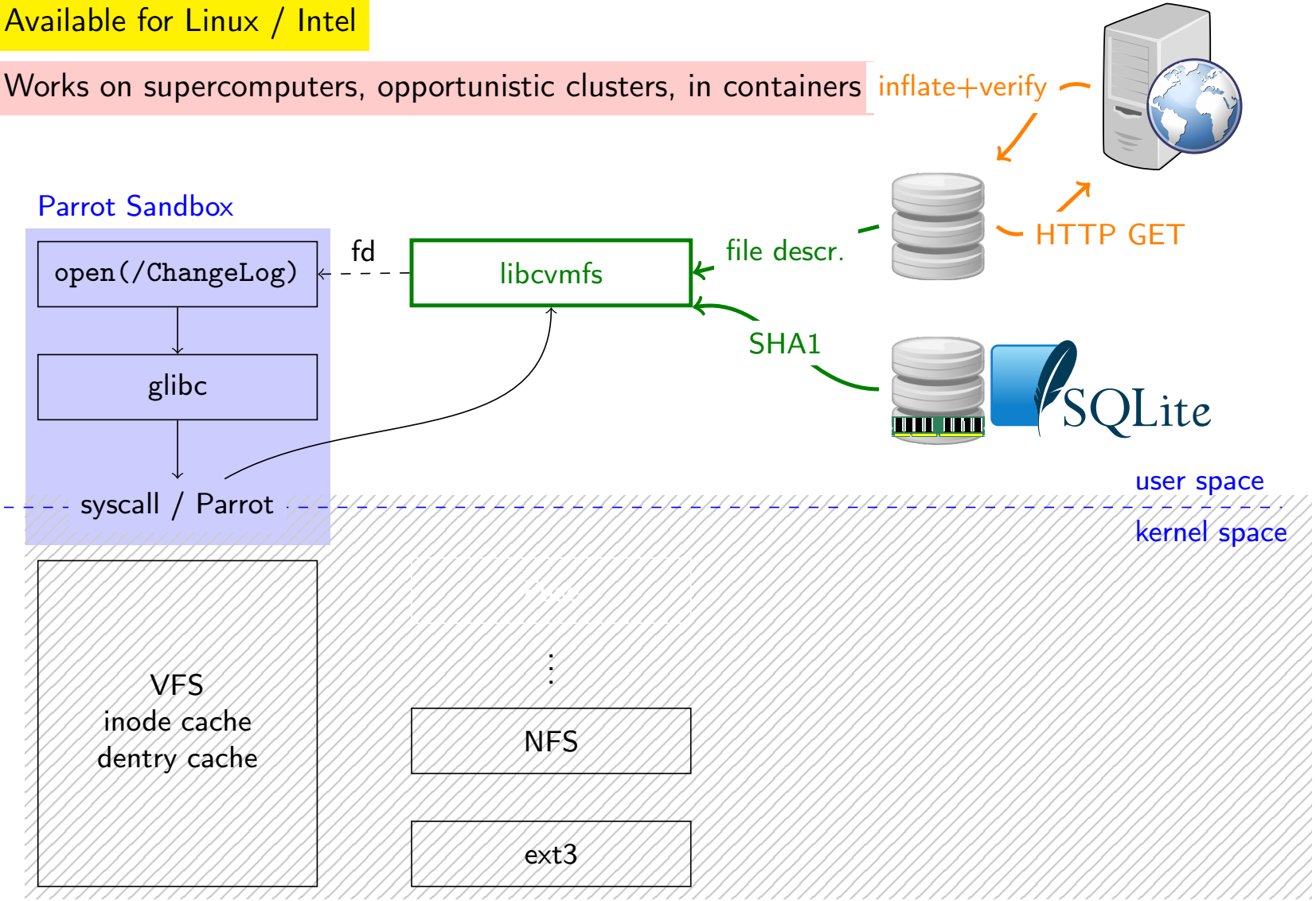
Works on most grids and virtual machines (cloud)



Mounting the File System Client: Parrot

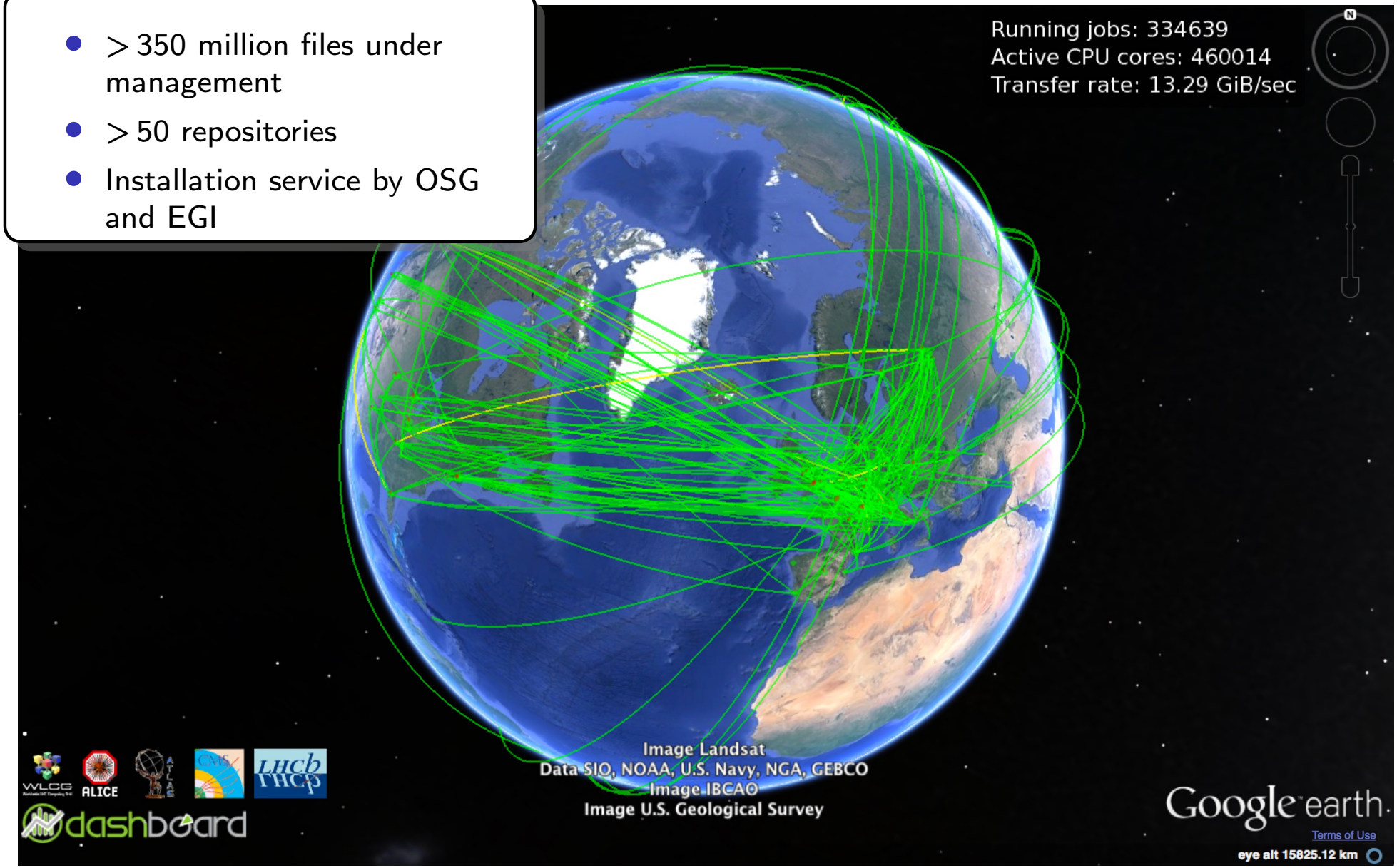
Available for Linux / Intel

Works on supercomputers, opportunistic clusters, in containers



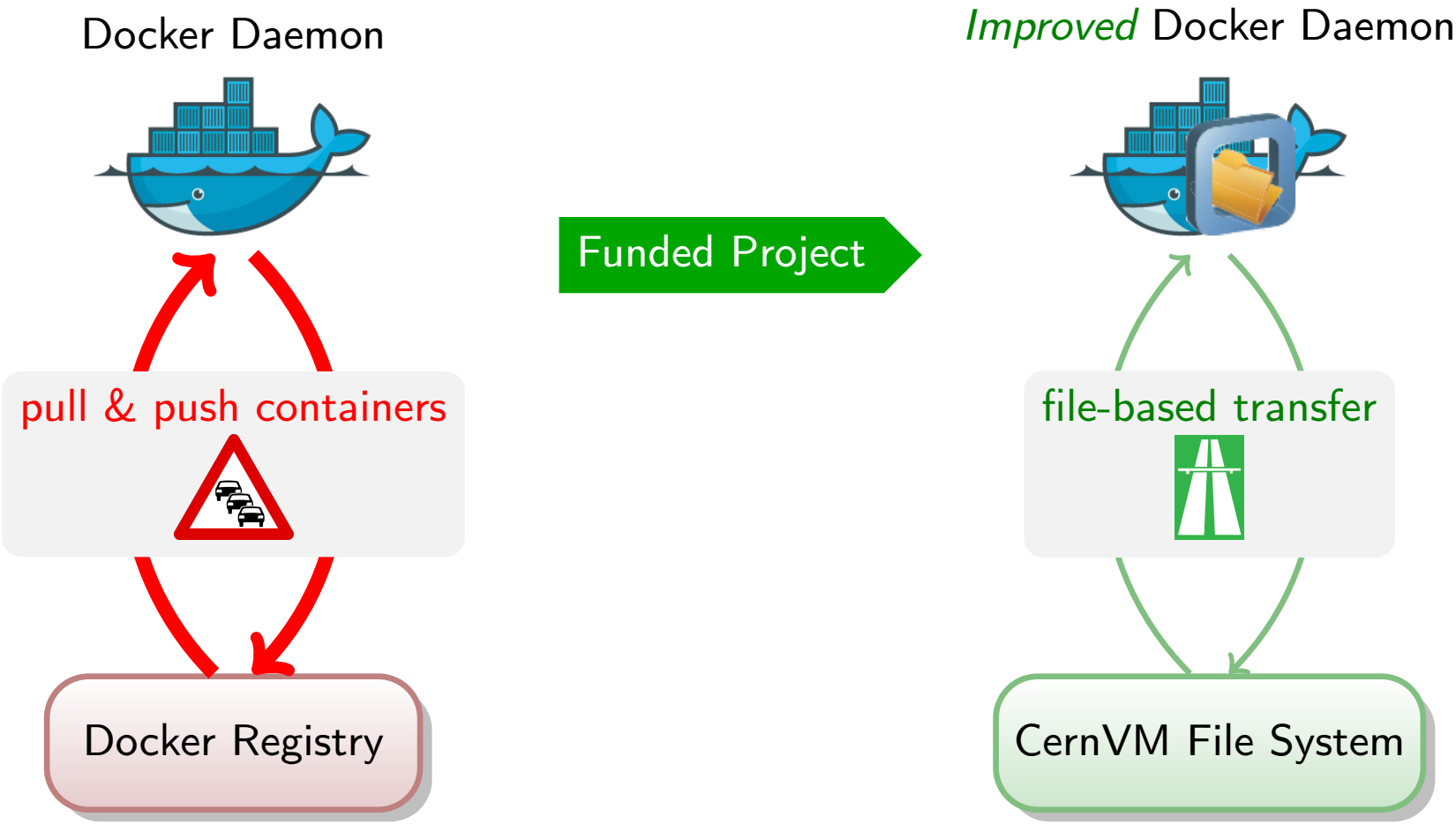
Scale of Deployment

- > 350 million files under management
- > 50 repositories
- Installation service by OSG and EGI



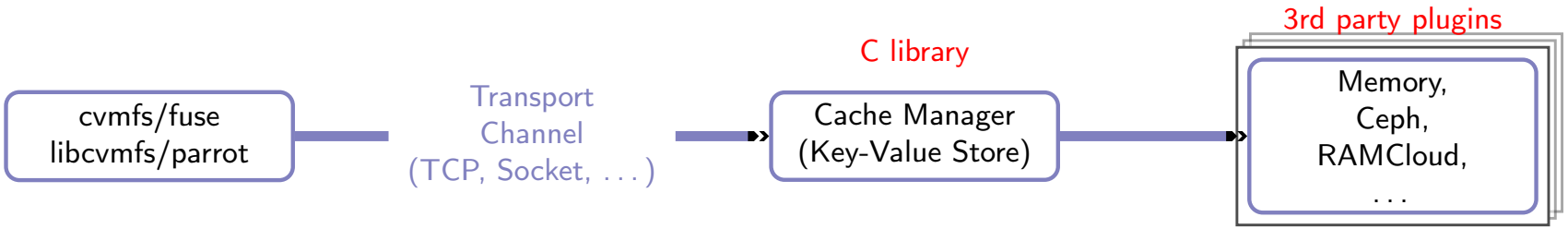
Docker Integration

Under Construction!



Client Cache Manager Plugins

Under Construction!



Draft C Interface

```
cvmfs_add_refcount(struct hash object_id, int change_by);
cvmfs_pread(struct hash object_id, int offset, int size,
            void *buffer);

// Transactional writing in fixed-sized chunks
cvmfs_start_txn(struct hash object_id, int txn_id,
                struct info object_info);
cvmfs_write_txn(int txn_id, void *buffer, int size);
cvmfs_abort_txn(int txn_id);
cvmfs_commit_txn(int txn_id);
```

Summary

CernVM-FS

- Global, HTTP-based file system for software distribution
- Works great with Parrot
- Optimized for small files, heavy meta-data workload
- Open source (BSD), used beyond high-energy physics

Use Cases

- Scientific software
- Distribution of static data e. g. conditions, calibration
- VM / container distribution cf. *CernVM*
- Building block for long-term data preservation

Source code:	https://github.com/cvmfs/cvmfs
Downloads:	https://cernvm.cern.ch/portal/filesystem/downloads
Documentation:	https://cvmfs.readthedocs.org
Mailing list:	cvmfs-talk@cern.ch

Backup Slides

CernVM-FS Client Tools

Fuse Module

- Normal namespace: `/cvmfs/<repository>` e. g. `/cvmfs/atlas.cern.ch`
- Private mount as a user possible
- One process per fuse module + watchdog process
- Cache on local disk
- Cache LRU managed
- NFS Export Mode
- Hotpach functionality `cvmfs_config reload`

Parrot

- Built in by default

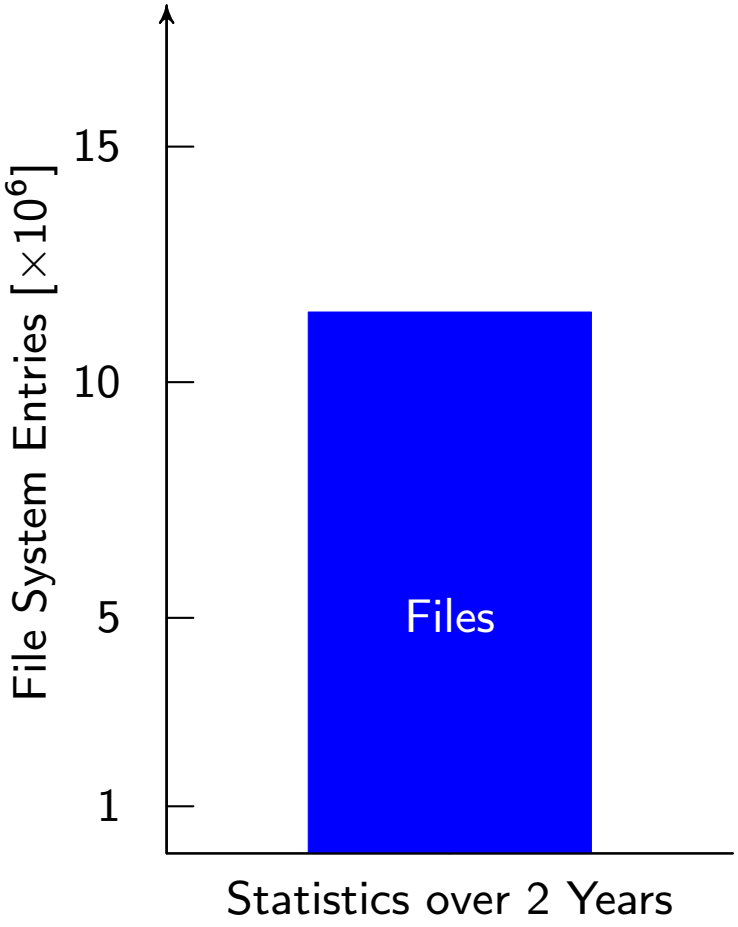
Mount helpers

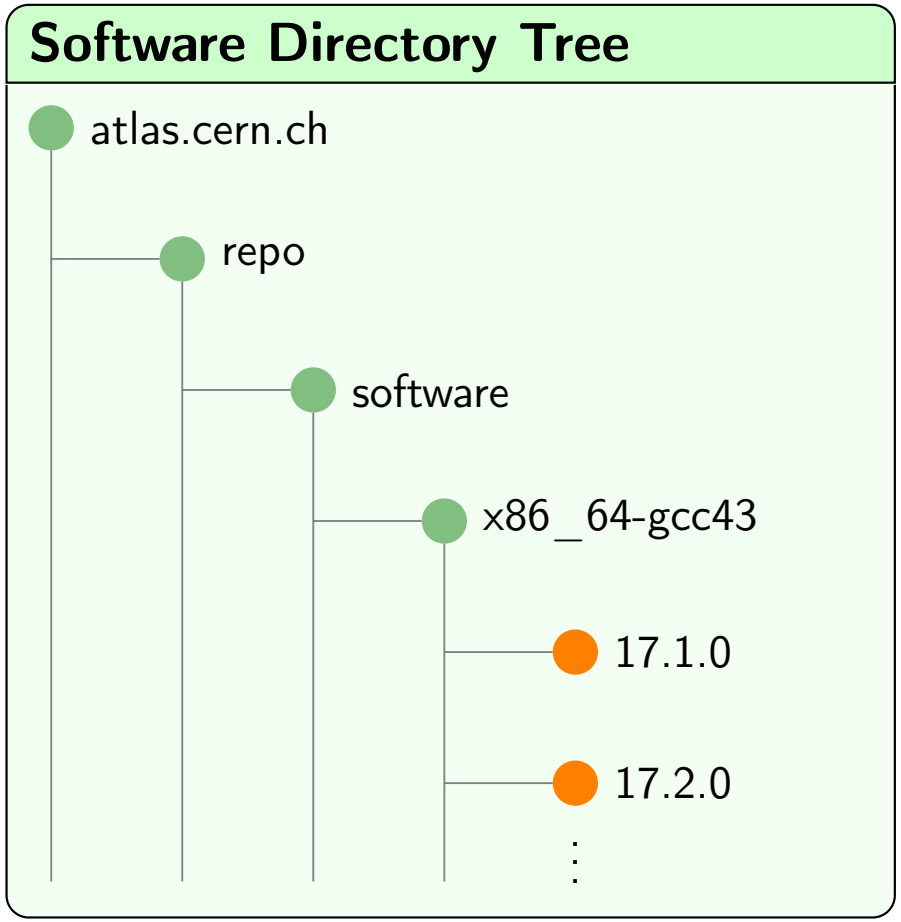
- Setup environment (number of file descriptors, access rights, . . .)
- Used by autofs on /cvmfs
- Used by /etc/fstab or mount as root
  `mount -t cvmfs atlas.cern.ch /cvmfs/atlas.cern.ch`

Diagnostics

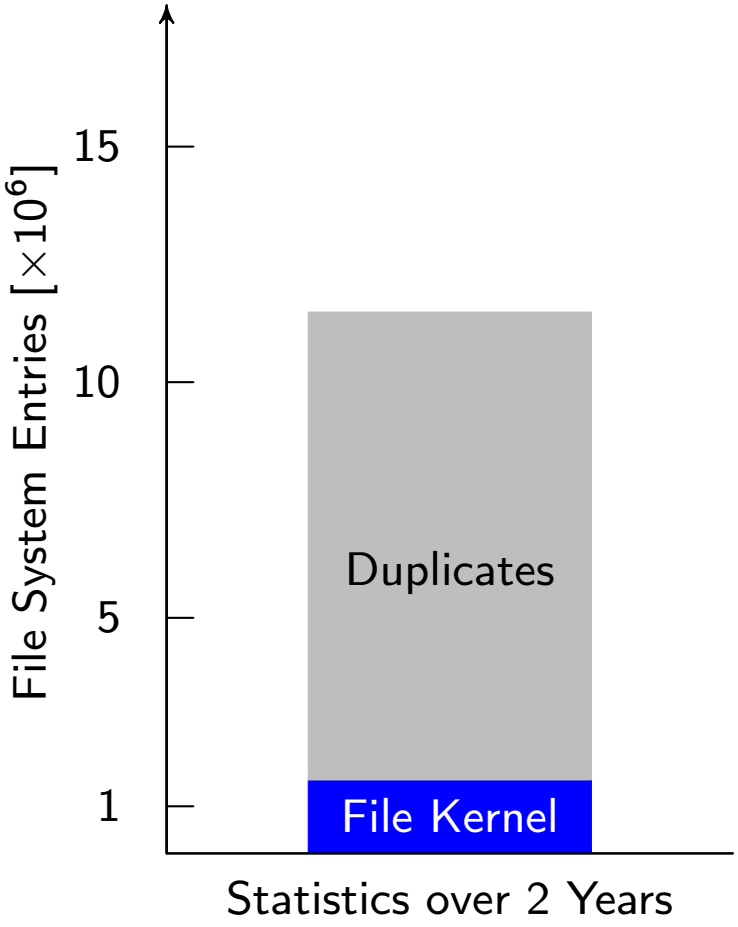
- Nagios check available
- `cvmfs_config probe`
- `cvmfs_config chksetup`
- `cvmfs_fsck`
- `cvmfs_talk`, connect to running instance

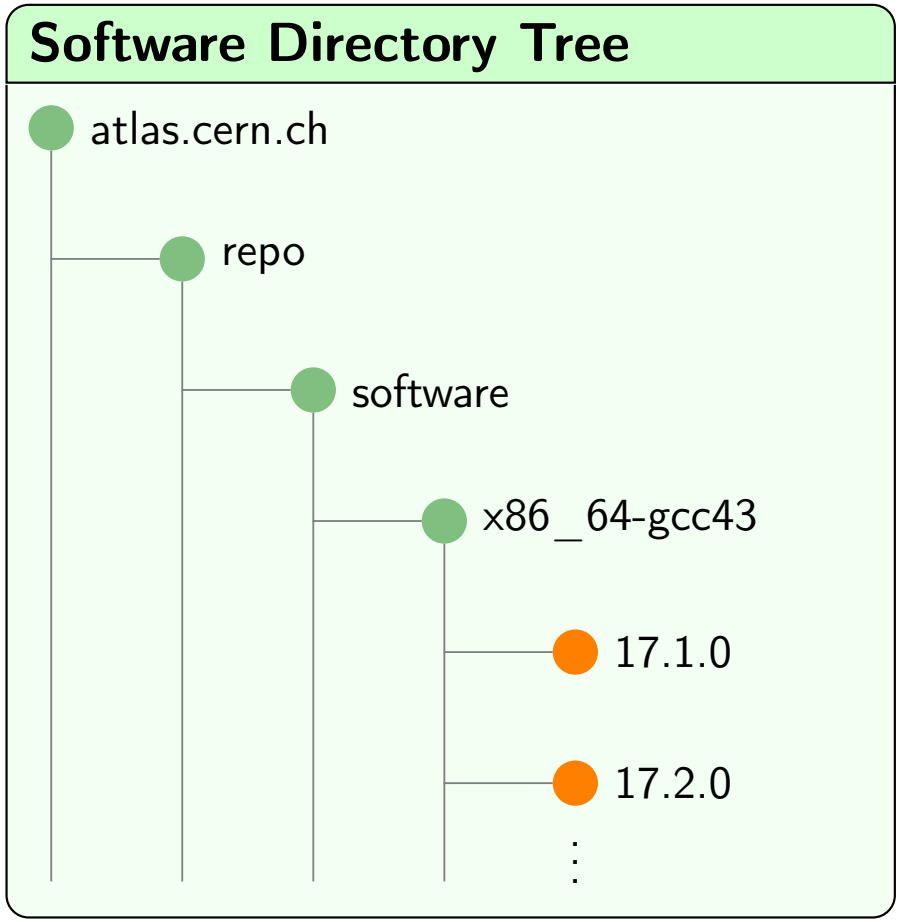
Experiment Software from a File System Viewpoint



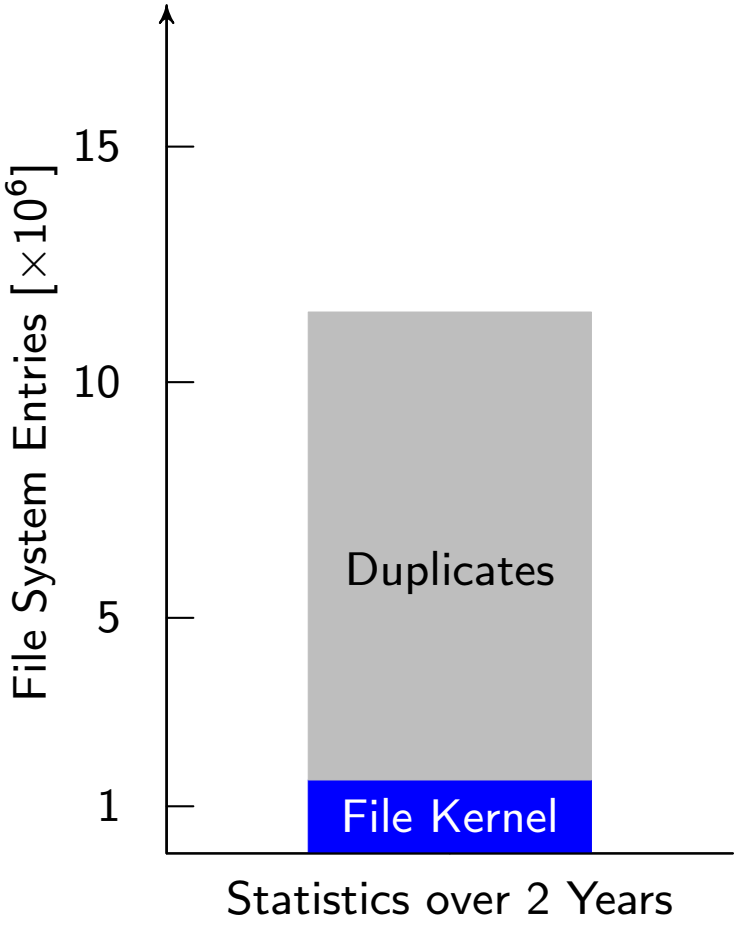


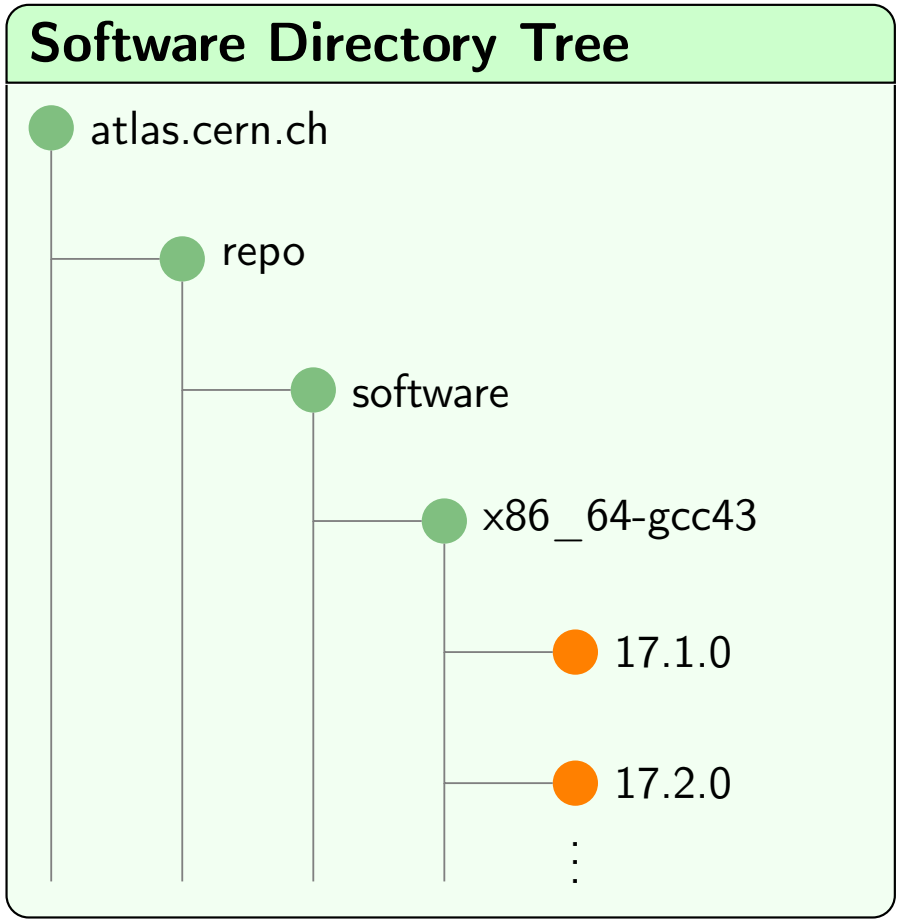
Experiment Software from a File System Viewpoint





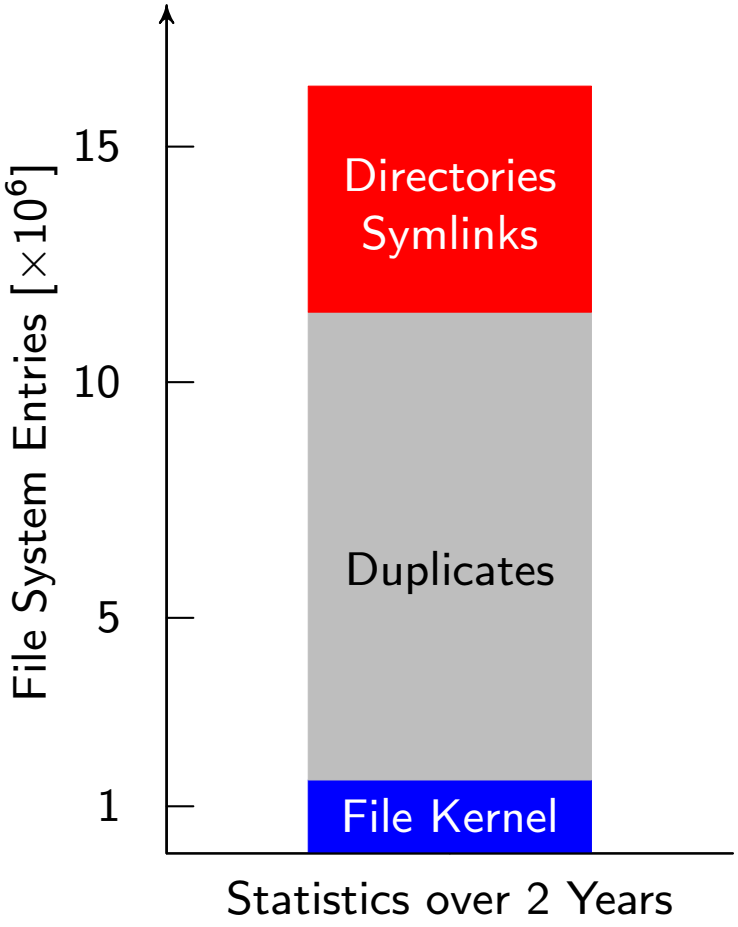
Experiment Software from a File System Viewpoint

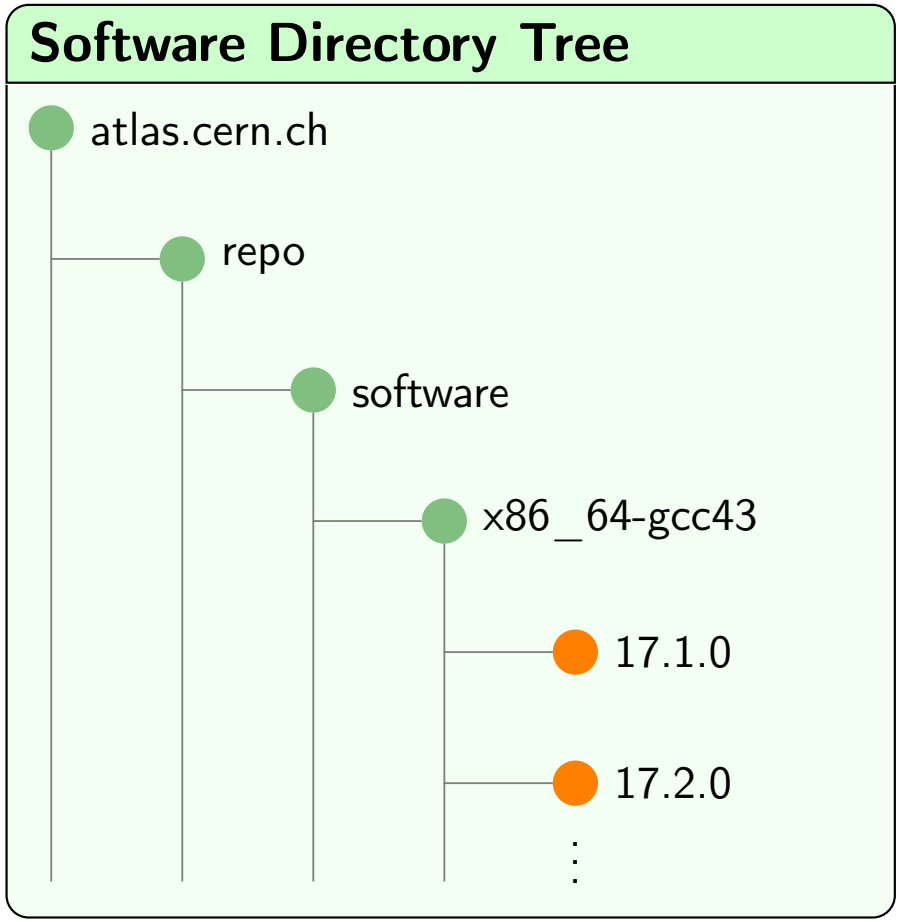




Between consecutive software versions: only ≈ 15 % new files

Experiment Software from a File System Viewpoint

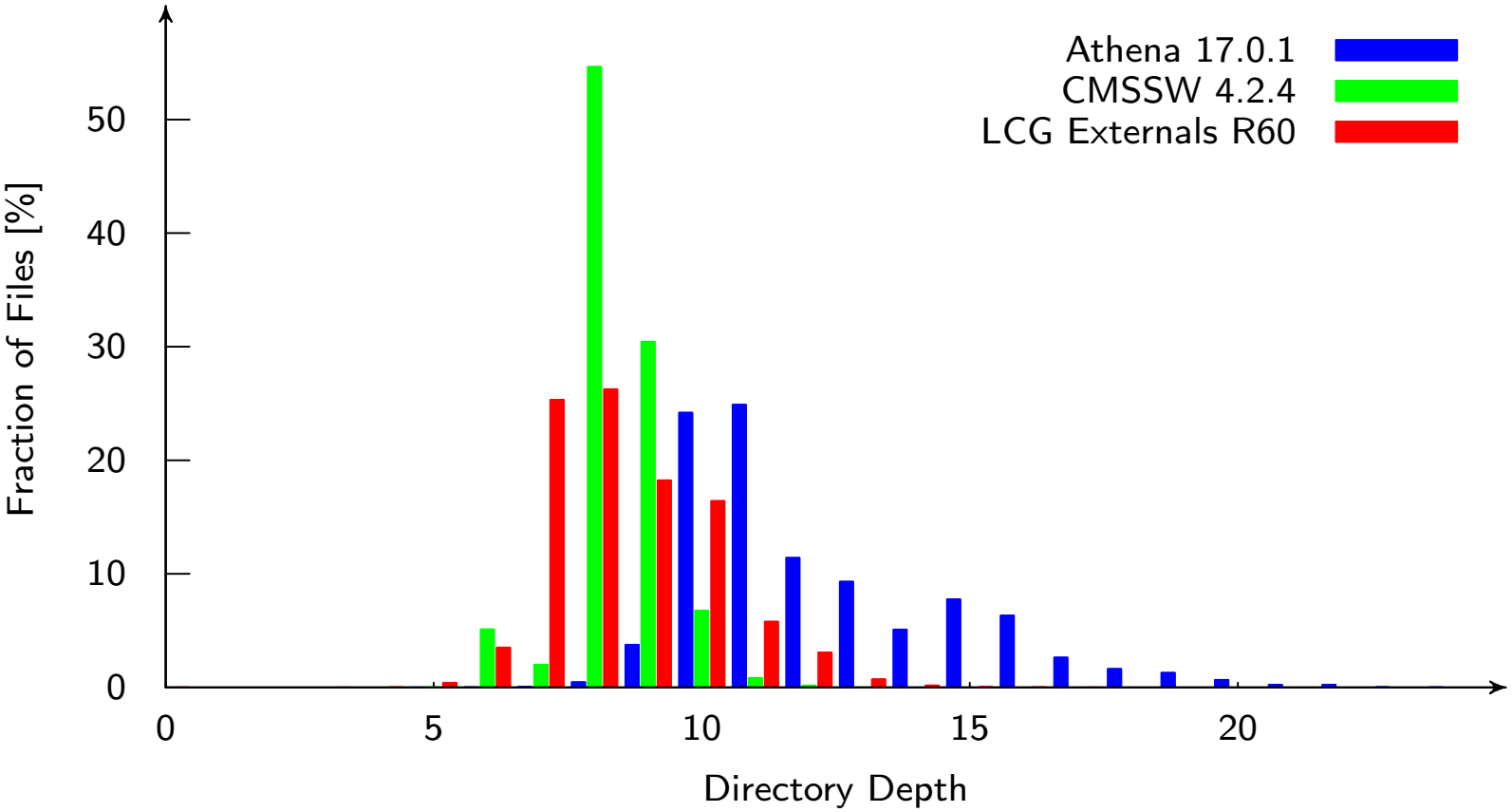




Fine-grained software structure (*Conway's law*)

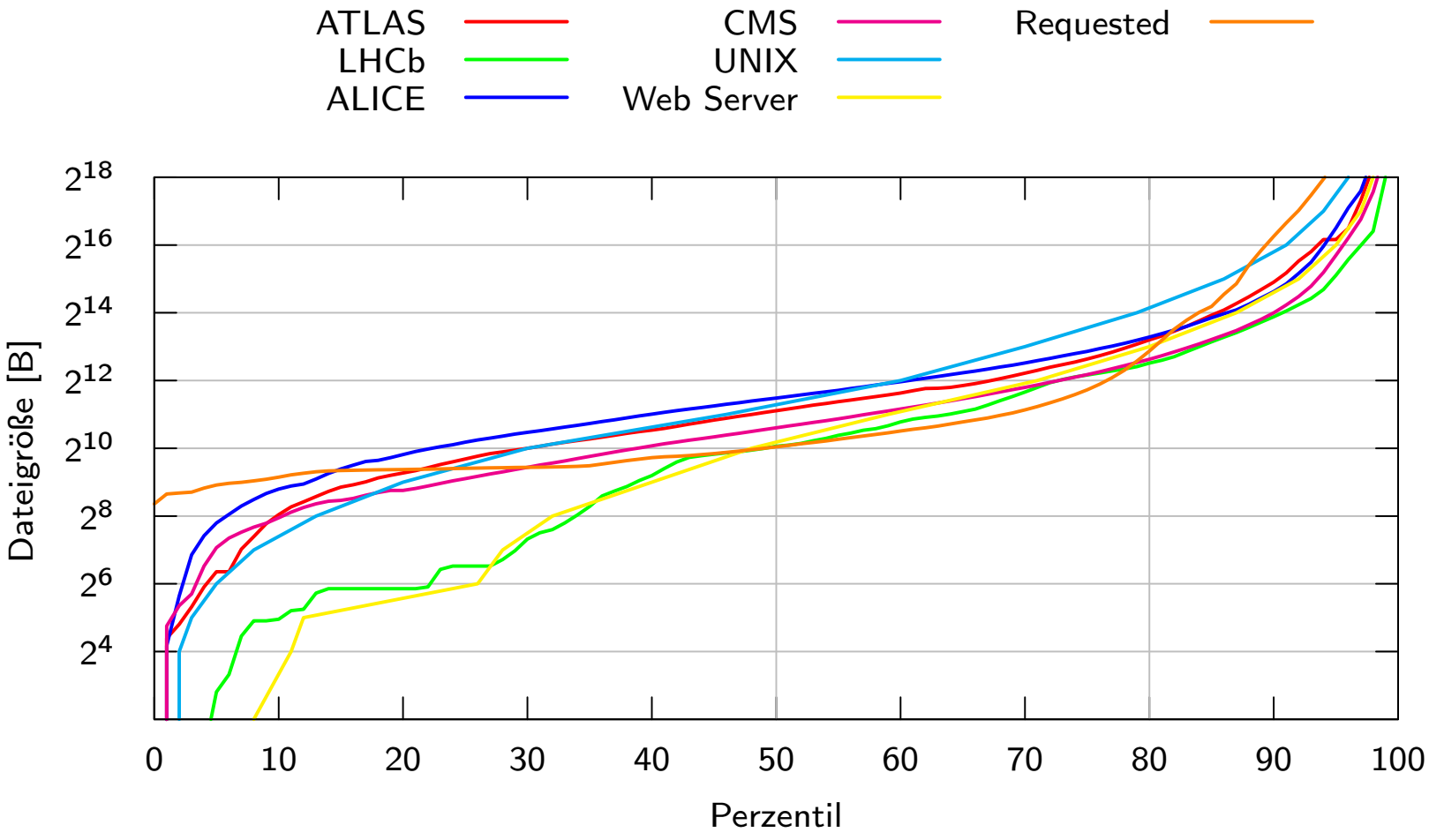
Between consecutive software versions: only ≈ 15 % new files

Directory Organization



Typical (non-LHC) software: majority of files in directory level ≤ 5

Cumulative File Size Distribution



cf. Tanenbaum et al. 2006 for "Unix" and "Webserver"

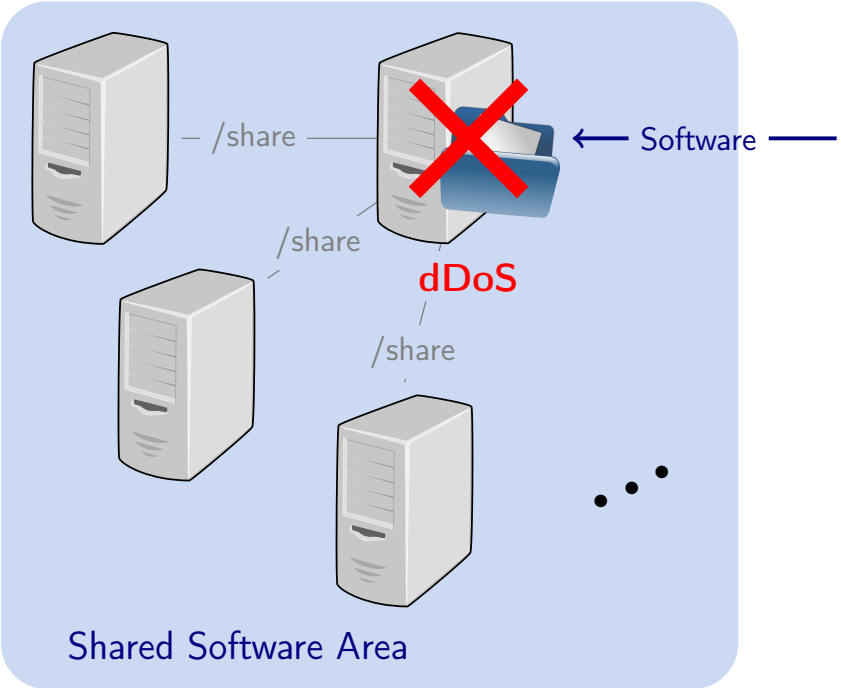
Good compression rates (factor 2–3)

Working Set

- ≈10 % of all available files are requested at runtime
- Median of file sizes: $< 4\,\text{kB}$

Flash Crowd Effect

- Up to 500 kHz meta data request rate
- Up to 1 kHz file open request rate



Based on ATLAS Figures 2012

Software	**Data**
POSIX Interface	put, get, seek, streaming
File dependencies	Independent files
10^7 objects	10^8 objects
10^{12} B volume	10^{16} B volume
Whole files	File chunks
Absolute paths	Any mountpoint
Open source	Confidential
WORM ("write-once-read-many")	
Versioned	