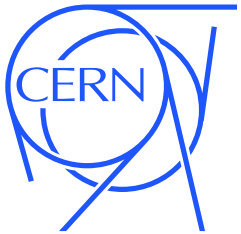


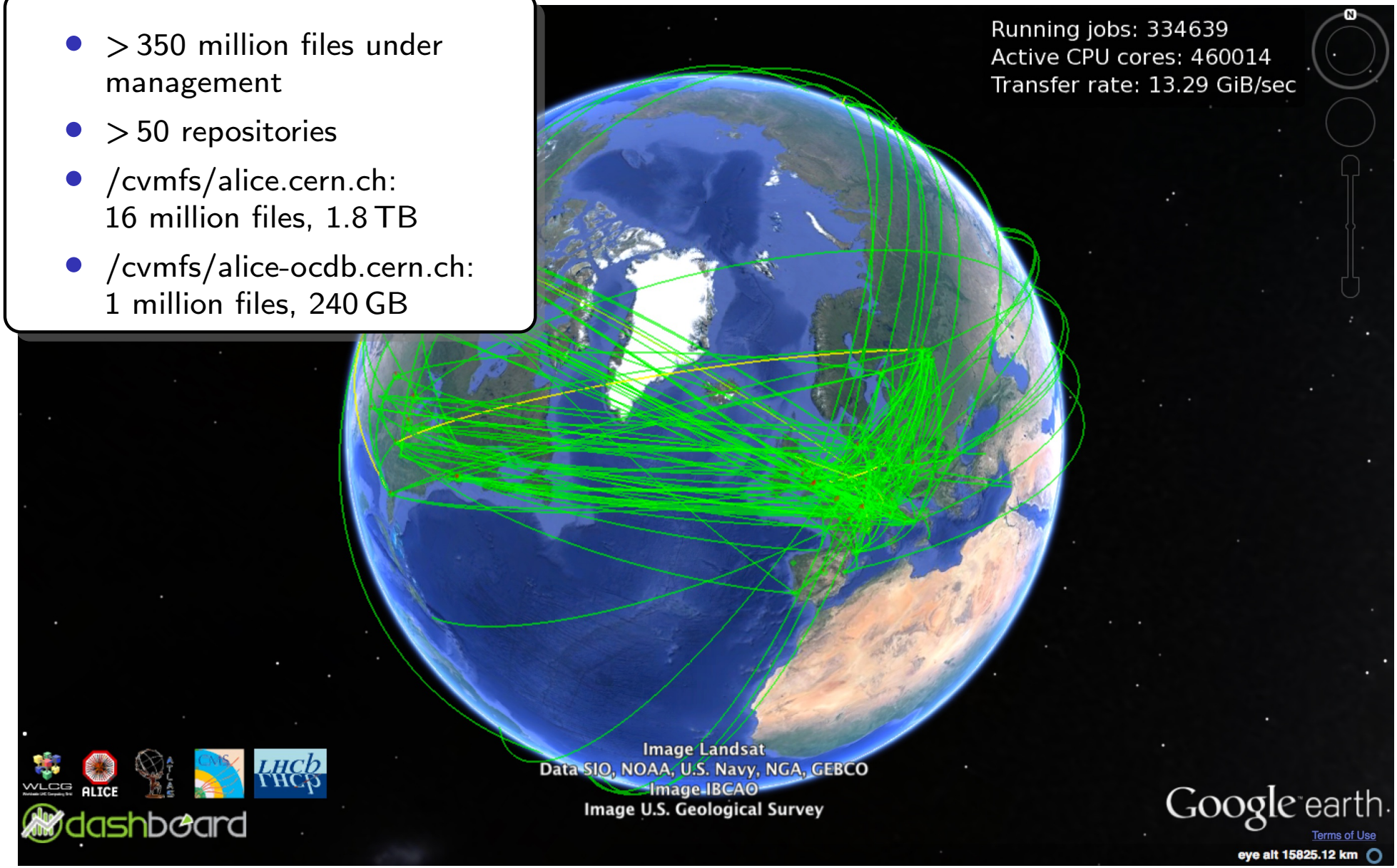
CernVM-FS News & Plans

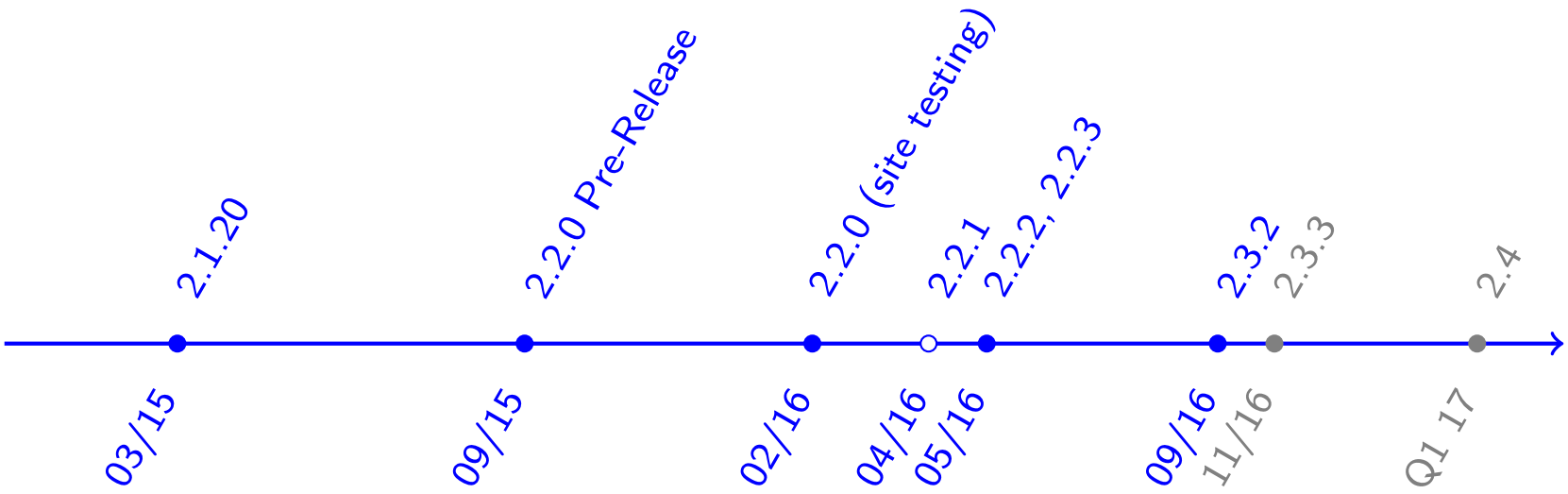
Jakob Blomer, Radu Popescu
CERN

ALICE Offline Week
November 2nd, 2016

Current Scale of Deployment

- > 350 million files under management
- > 50 repositories
- /cvmfs/alice.cern.ch:
  16 million files, 1.8 TB
- /cvmfs/alice-ocdb.cern.ch:
  1 million files, 240 GB





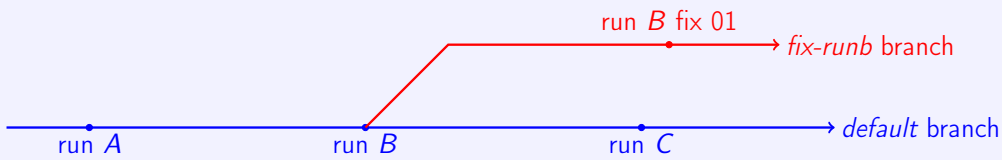
New features under construction for release 2.4

- Experimental REST API for writing ⟵ most of today's topic
- Instant access to snapshots
- Client cache plugins

Developments Under Construction

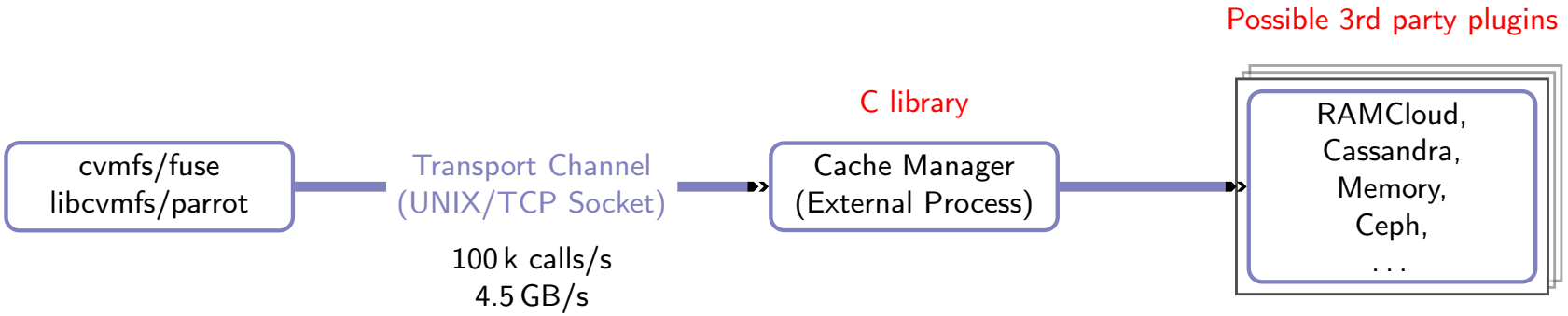
New features under construction

❶ **Branching.** Use of named file system snapshots that link conditions data to LHC run periods



❷ **Instant Access to Snapshots.** A new virtual directory that provides access snapshot access within a single mountpoint, like `/cvmfs/alice-ocdb.cern.ch/.cvmfs_snapshots/runXYZ`

CernVM-FS Cache Plugins



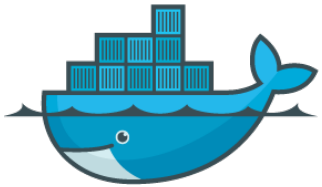
Motivation for cache plugins (continuation of Tim Shaffer's work)

- More **flexibility** for client deployment:
  - Diskless server farms
  - HPC "burst buffers": utilize fast, possibly non-POSIX storage
- Opens the door to external contributions!

For standard deployment on the Grid nothing changes!

Up for mid 2017

Docker Daemon



pull & push containers

Docker Registry

12 Months Project

Improved Docker Daemon

file-based transfer

CernVM File System

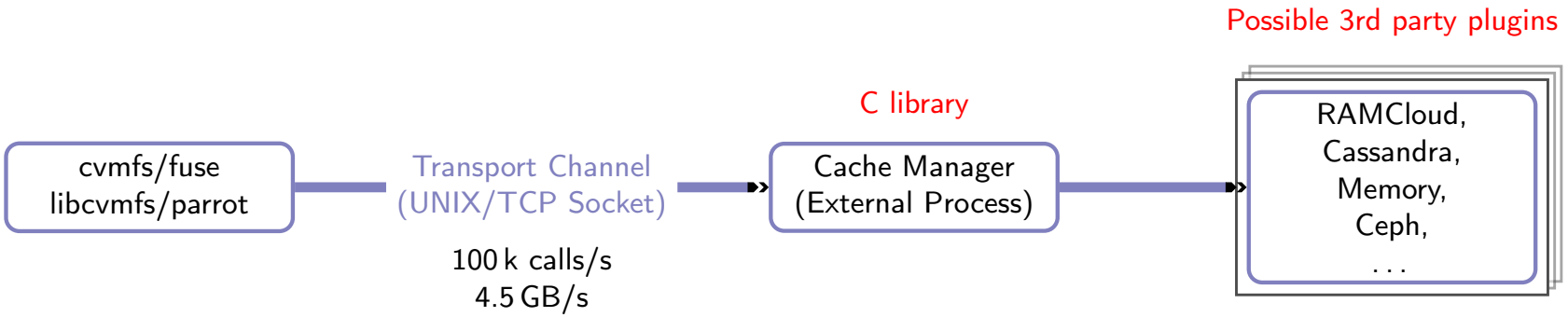
Improvements for Writing into CernVM-FS

Backup Slides

CernVM-FS Cache Plugin C Interface

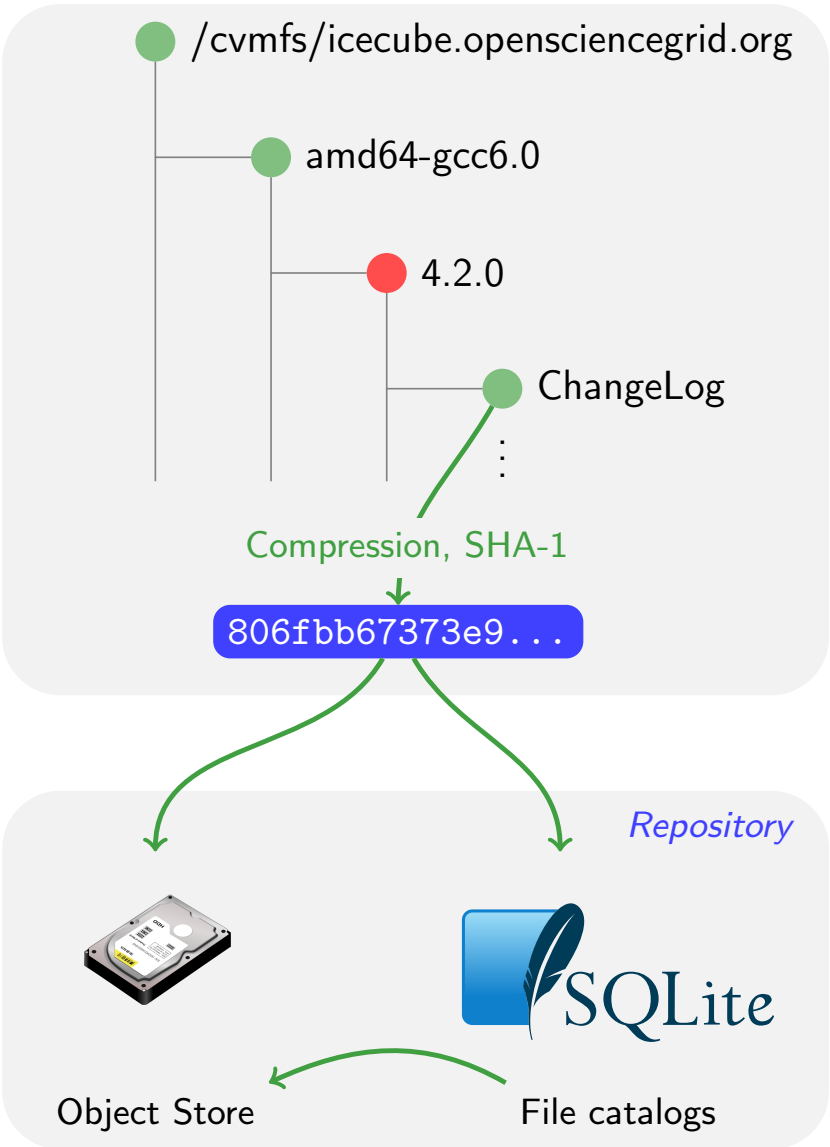
Callbacks to be implemented by plugin developer

```
// Reading data
int cvmcache_chrefcnt(struct hash object_id, int change_by);
int cvmcache_object_info(struct hash object_id,
                         struct object_info *info);
int cvmcache_pread(struct hash object_id,
                   int offset, int size,
                   void *buffer);

// Transactional writing in fixed-sized parts
int cvmcache_start_txn(struct hash object_id, int txn_id,
                       struct info object_info);
int cvmcache_write_txn(int txn_id, void *buffer, int size);
int cvmcache_abort_txn(int txn_id);
int cvmcache_commit_txn(int txn_id);

// Optional: quota management
int cvmcache_shrink(int shrink_to, int *used);
int cvmcache_listing_begin(int lst_id, ...);
int cvmcache_listing_next(int lst_id, ...);
int cvmcache_listing_end(int lst_id);
```



- ✓ Protocol definition in Google protobuf
- ✓ Socket and transport handling
- ✓ Plugin C library
- ✓ Client-side unit tests (white box)
- ✓ Unit tests for plugins (black box)
- ✓ Demo plugin storing data in `std::string`
- ✓ RAMCloud plugin
- Cache configuration syntax
- Client hotpatch support

Content-Addressable Storage: Data Structures



Object Store

- Compressed files and chunks
- De-duplicated

File Catalog

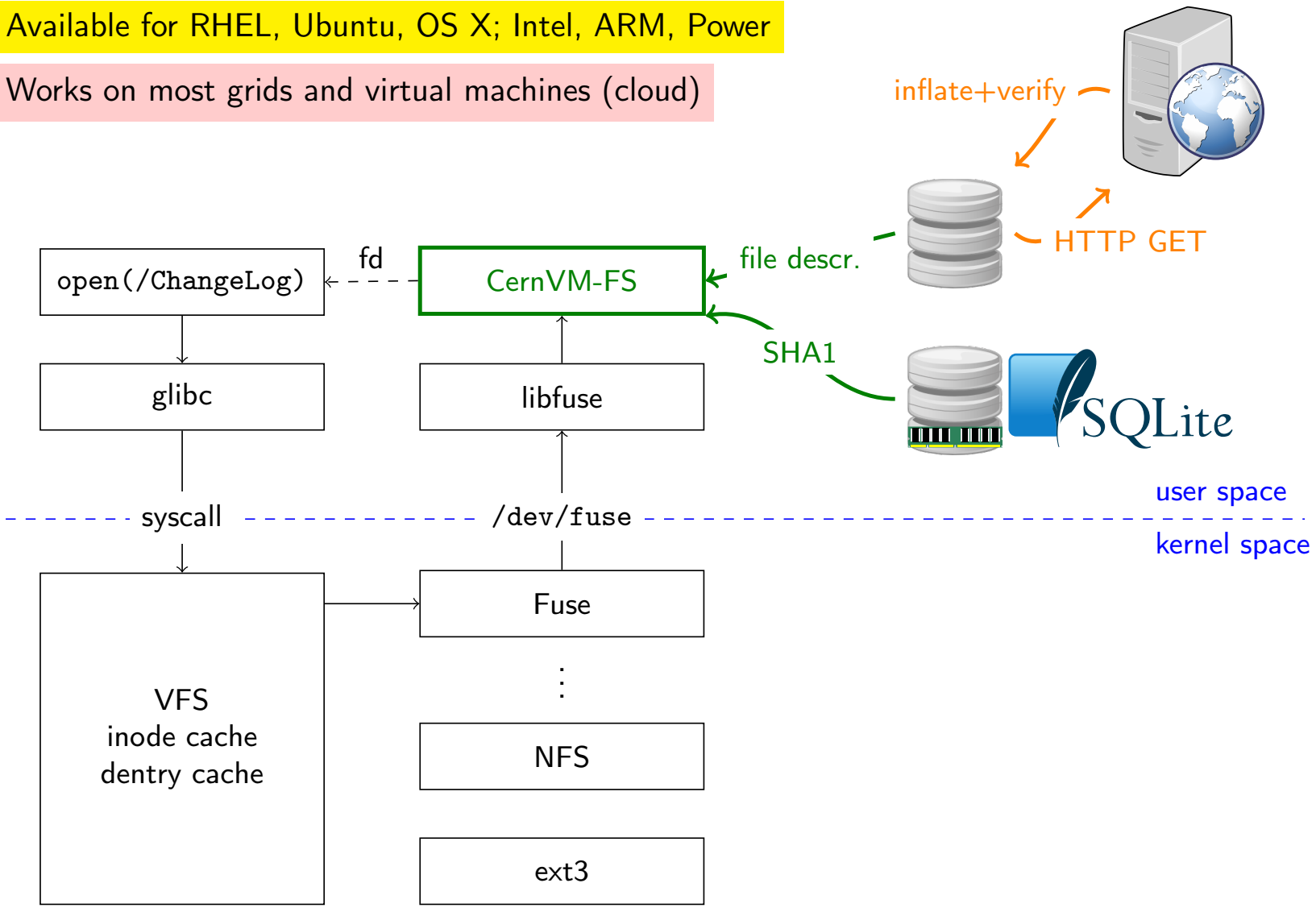
- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed
  $\Rightarrow$ integrity, authenticity
- Time to live
- Partitioned / Merkle hashes
  (possibility of sub catalogs)

$\Rightarrow$ **Immutable files, trivial to check for corruption, versioning**

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