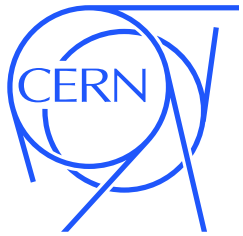
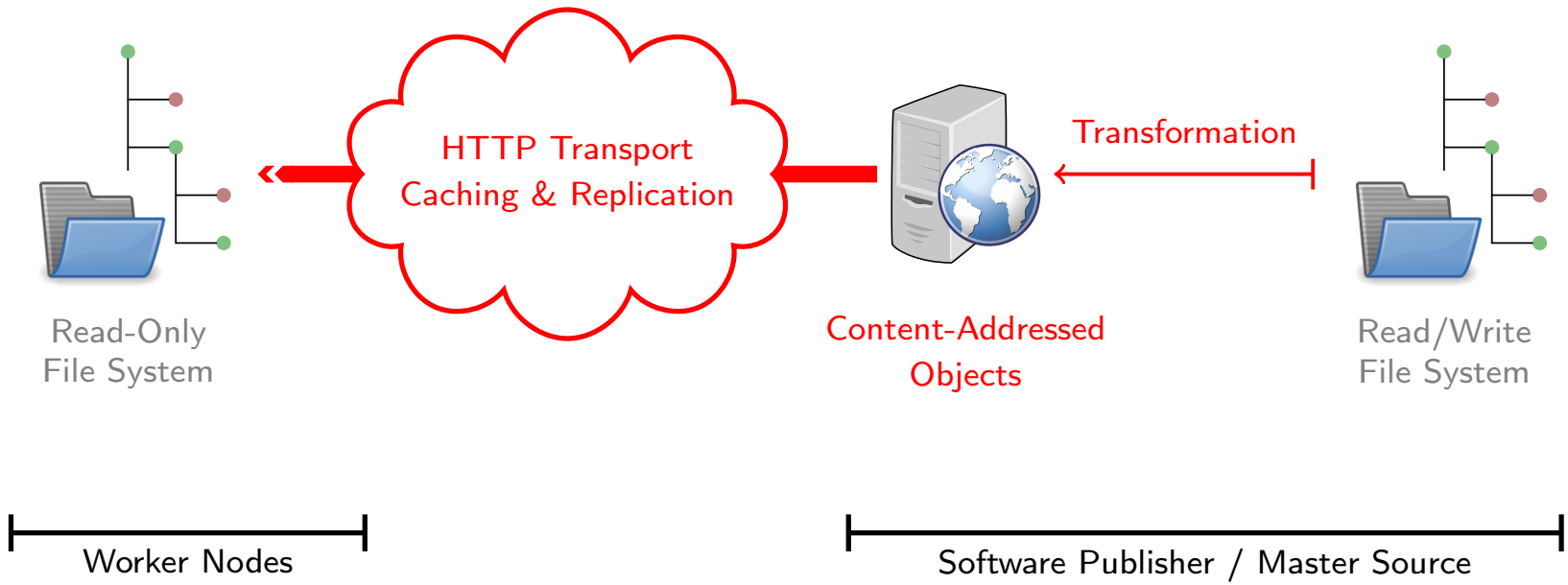


Cache Plugins for CernVM-FS: a Proof-of-Concept with RAMCloud

Jakob Blomer

CERN, SFT Group Meeting
October 31st, 2016

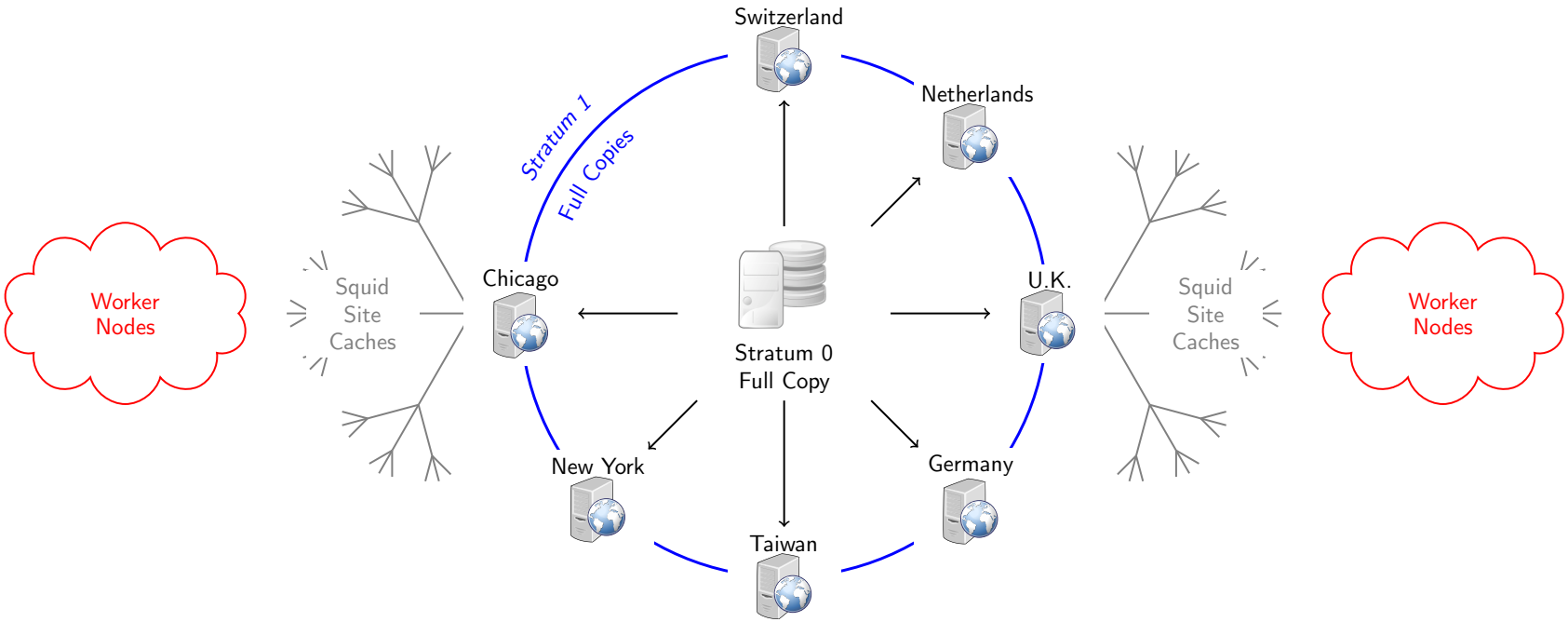
CernVM-FS Conceptual View



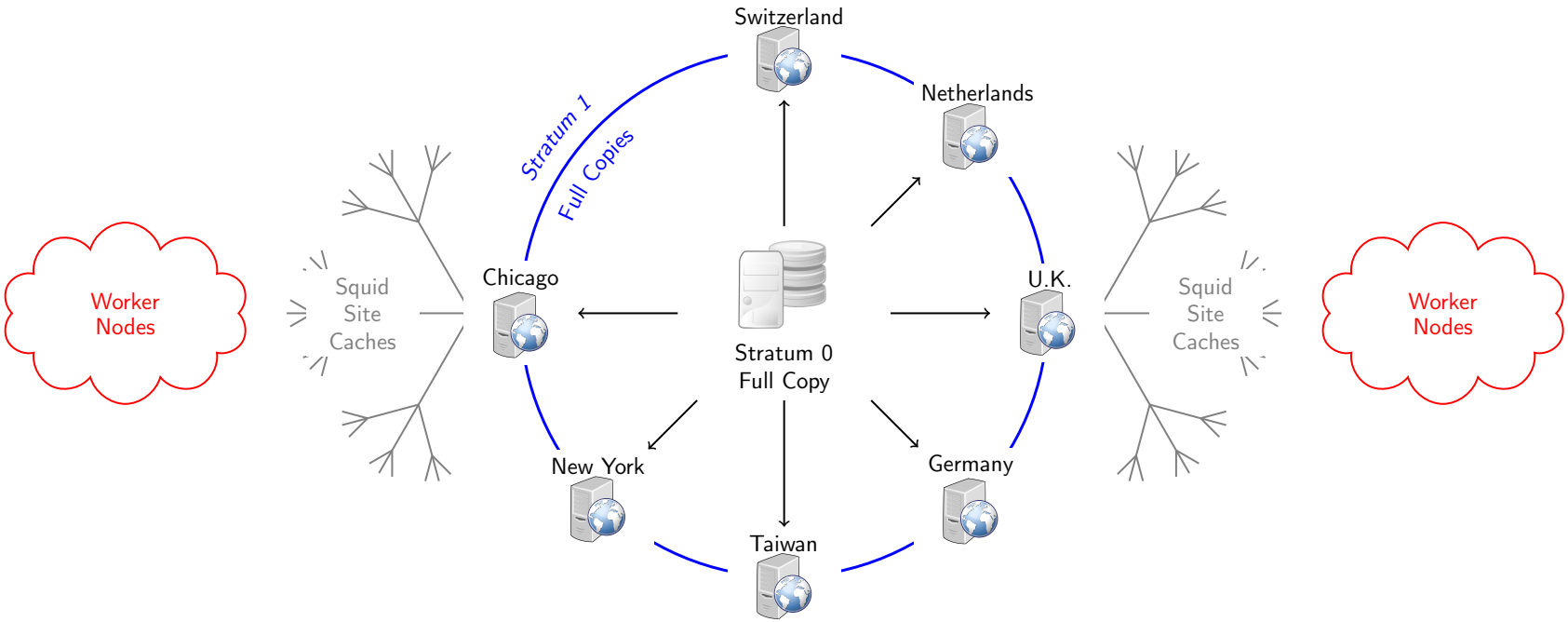
Two independent issues

1. How to distribute **and cache** small (1 kB – 50 MB), immutable objects?
2. How to present the objects as a mountable file system?

Distribution and Caching in CernVM-FS

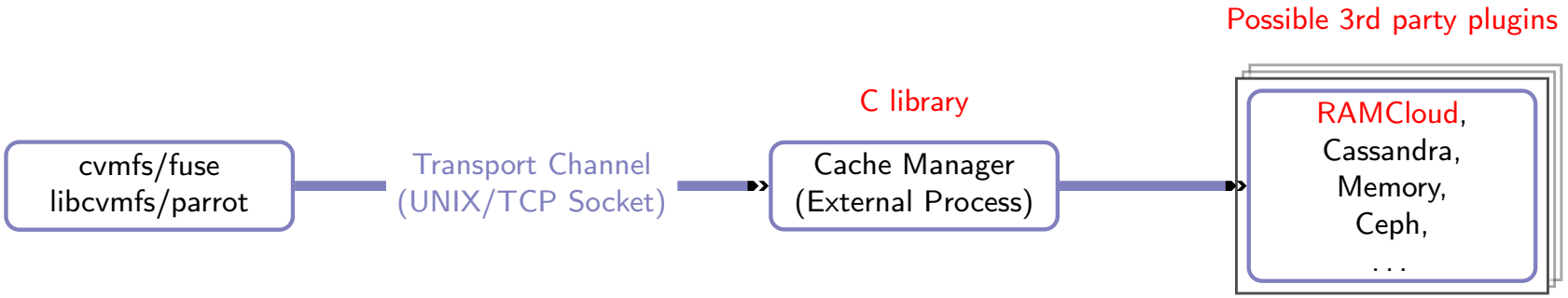


- Full Data Copies: Stratum 0, Stratum 1
- Sites' hot set: Squid web caches
- Worker nodes' hot set: local `/var/lib/cvmfs`



to be optionally replaced by "cache plugins"

- Full Data Copies: Stratum 0, Stratum 1
- Sites' hot set: Squid web caches
- Worker nodes' hot set: local `/var/lib/cvmfs`



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
- CMS and LIGO started using CernVM-FS for **data distribution**
  → we need a site-wide cache pool to prevent uncontrolled replication of (large) data sets in local caches

Motivation for RAMCloud proof-of-concept

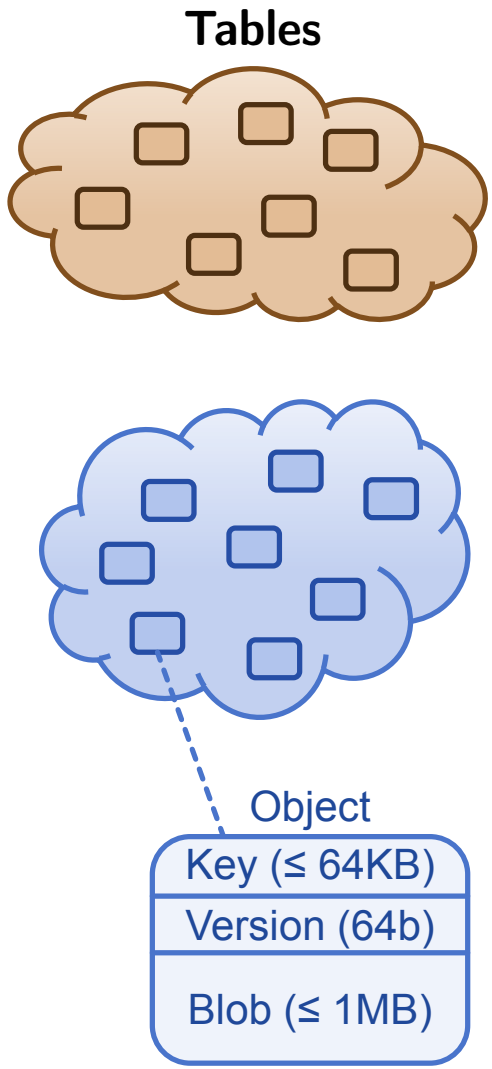
- Precursor for future data center storage platforms
- Developed by Stanford Computer Science, next door to CHEP'16, was excellent opportunity for discussion

Entities

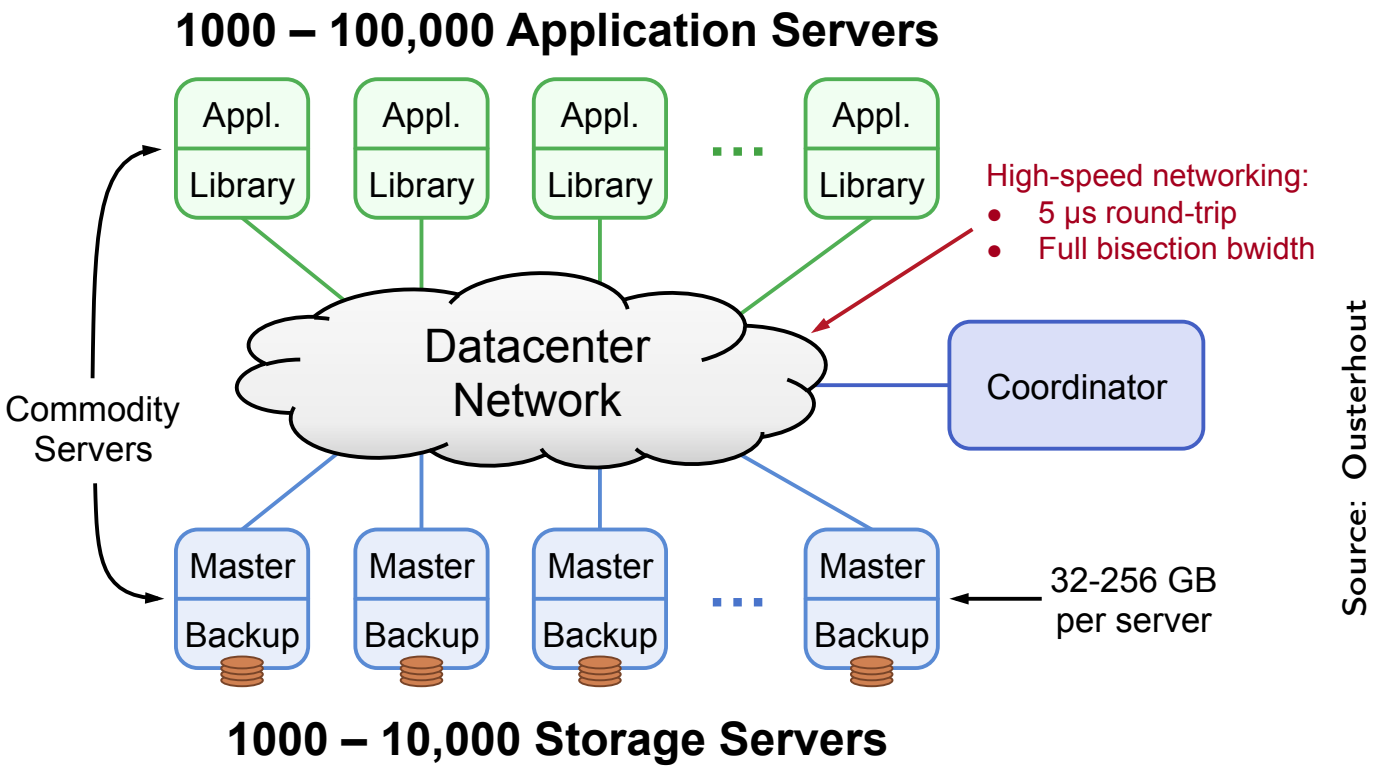
- Table
- Object (row): Key + Value + Version
- Tablet: partition of a table (block of rows)

Operations

- Read, write, delete single objects
- Conditional write, atomic increment
- Multi-object transactions
- Table enumeratation
- Secondary indices and range queries



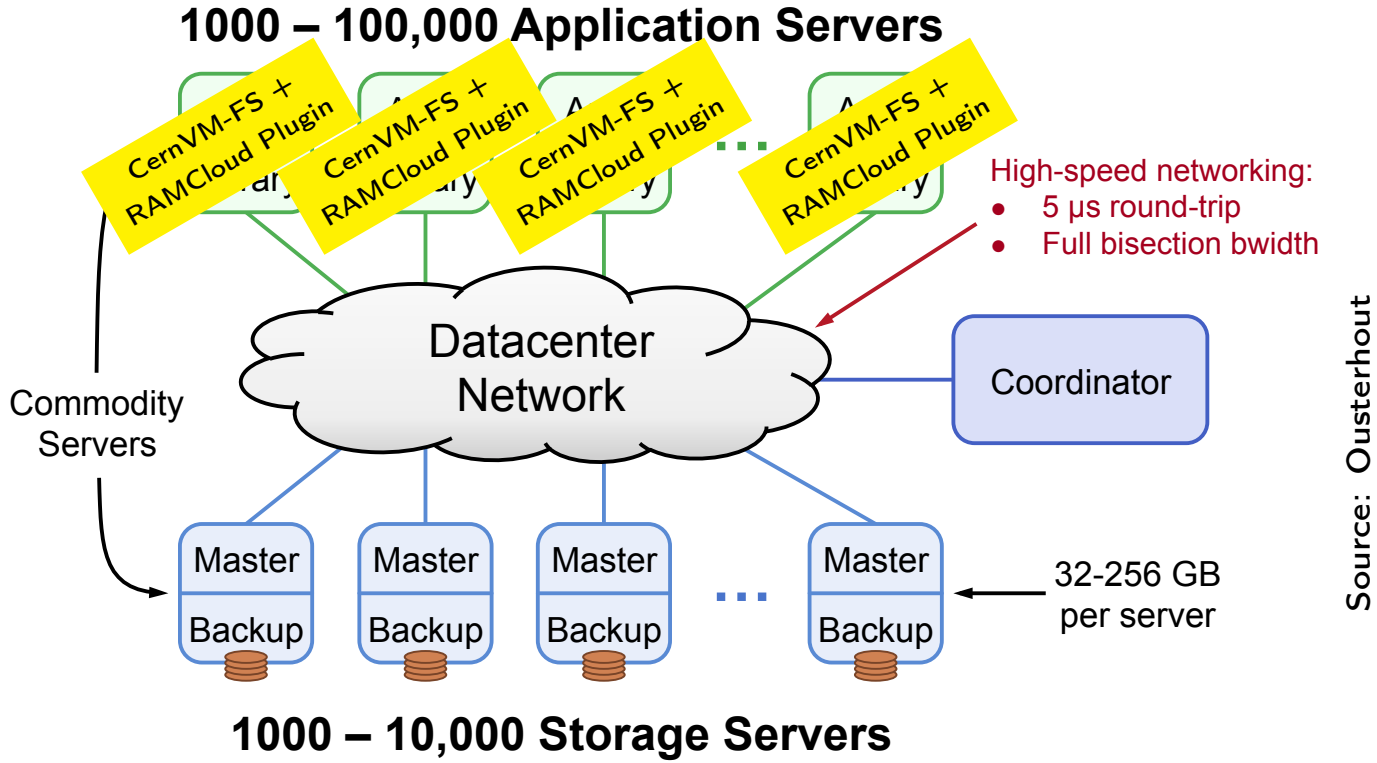
Source: Ousterhout



Key Properties

- Consistent, distributed key-value store with indexes
- All data guaranteed to be in **memory**, thus up to 1M ops/sec/server
- Reliable, *k* **replicas on disk** (buffered log, no disk write during store)
- Extra **low latency** (InfiniBand): 5 µs to read, 15 µs to write

RAMCloud System Overview



Key Properties

- Consistent, distributed key-value store with indexes
- All data guaranteed to be in **memory**, thus up to 1M ops/sec/server
- Reliable, *k* **replicas on disk** (buffered log, no disk write during store)
- Extra **low latency** (InfiniBand): 5 µs to read, 15 µs to write

CernVM-FS Cache Plugin C Interface

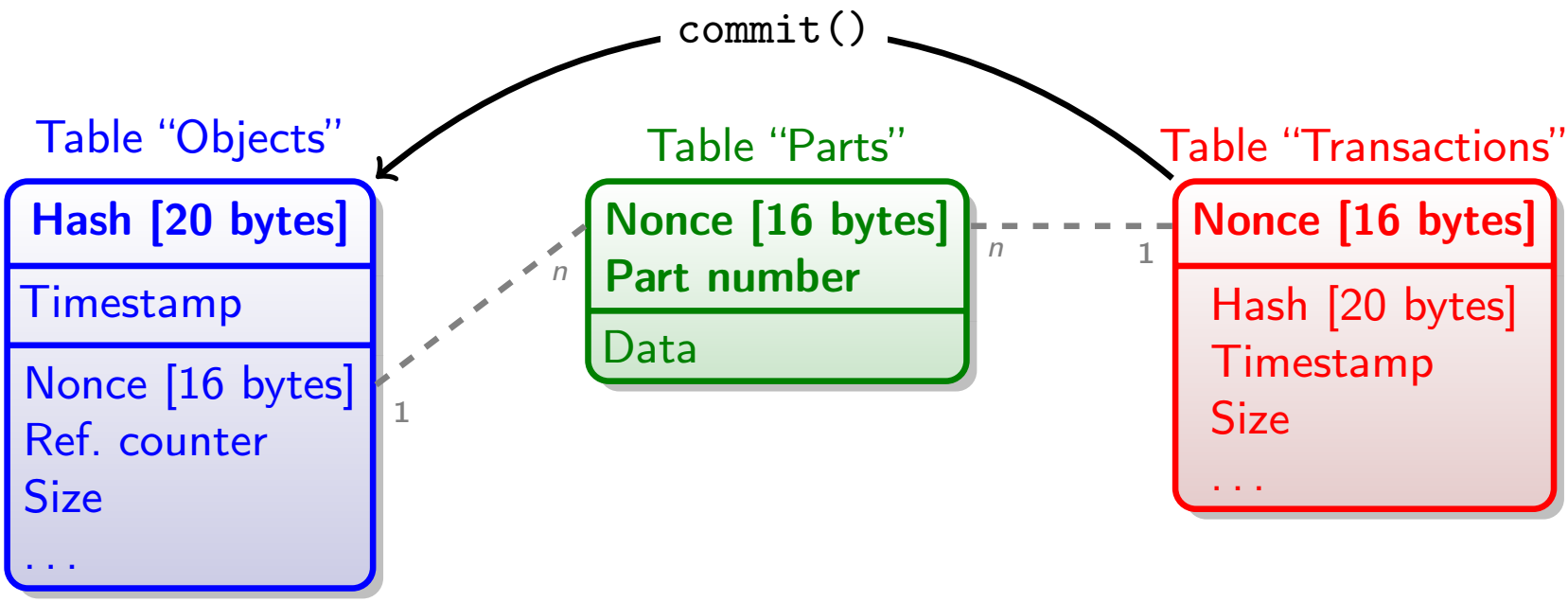
Callbacks to be implemented by plugin developer

```
// 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 cvmcache_listing_next(int listing_id, ...);
int cvmcache_listing_end(int listing_id);
```

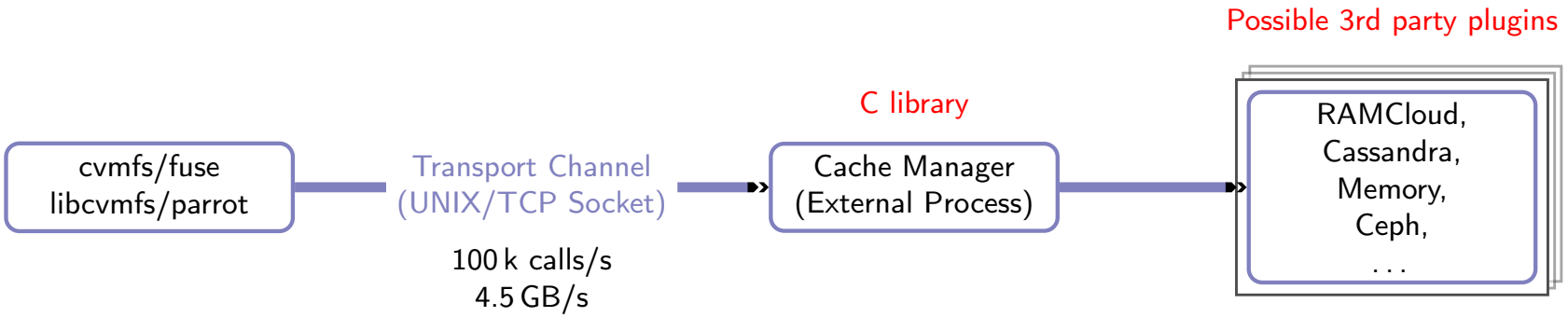
Data Structures of the Cache Plugin on RAMCloud



Special techniques

1. Clients regularly renew timestamp of objects & transactions
   $\longrightarrow$ Allows for garbage collection from crashed clients
2. Identify data blocks by *nonce* (number used once)
   $\longrightarrow$ Simplifies races of multiple clients on same object

Development Status



- ✓ 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

Summary

Cache Plugins

- Important steps towards **non-standard deployments** of the client such as in HPC environments and for data distribution use cases
- Opens the door to **external contributions**
- External cache manager will probably be used in a **tiered manner**: A small upper local cache in conjunction with a large lower cluster cache with relaxed semantics (no reference counting necessary)

Useful utility: Google Micro-Benchmarks

- Google microbenchmarks: open source library for benchmarking short-running code paths
- Usage similar to Google test
- Library knows how to measure time, how often to repeat a code snippet, CPU frequency scaling, ...
- Very useful to gather facts and to benchmark new platforms
- https://github.com/google/benchmark

Off topic: cctools'16 workshop

Cooperative Computing Tools Workshop
on Scalable Scientific Computing Oct 19-20 2016
http://ccl.cse.nd.edu/workshop/2016/

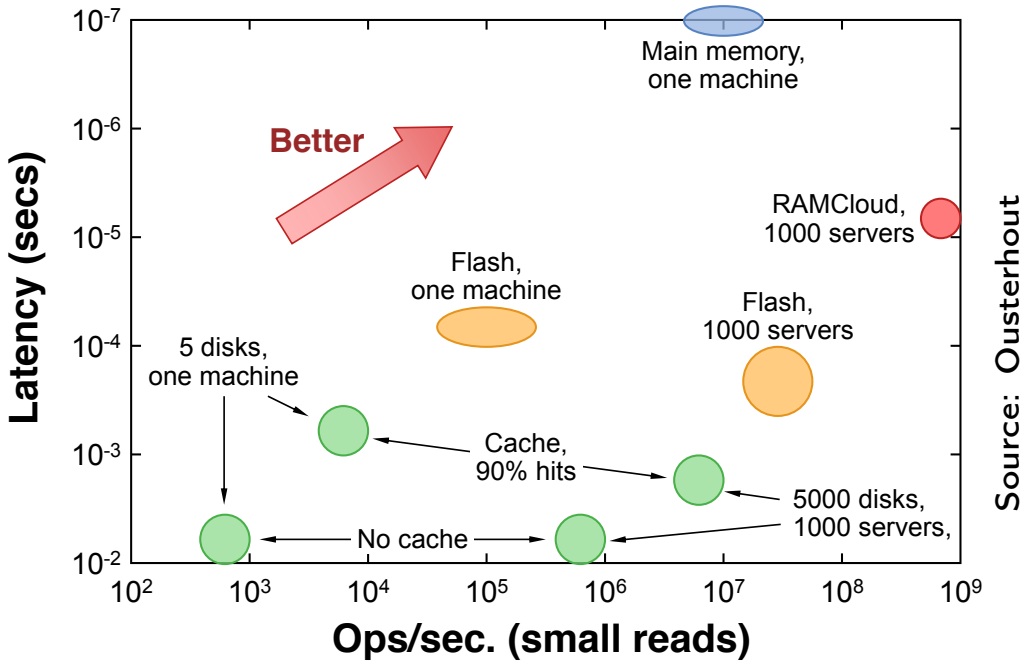
- Participation from multiple scientific domains,
  e. g. high-energy physics, astro physics, bio informatics, ...
- Presentation on "Global Software Distribution with CernVM-FS"
- Many interesting tools presented:
  - *Parrot* (used by CernVM-FS)
  - *Workqueue* and *Makeflow* (used by ALICE)
  - *Umbrella* (software preservation, used by some CMS groups)

We should seize opportunities to collaborate with computer science groups

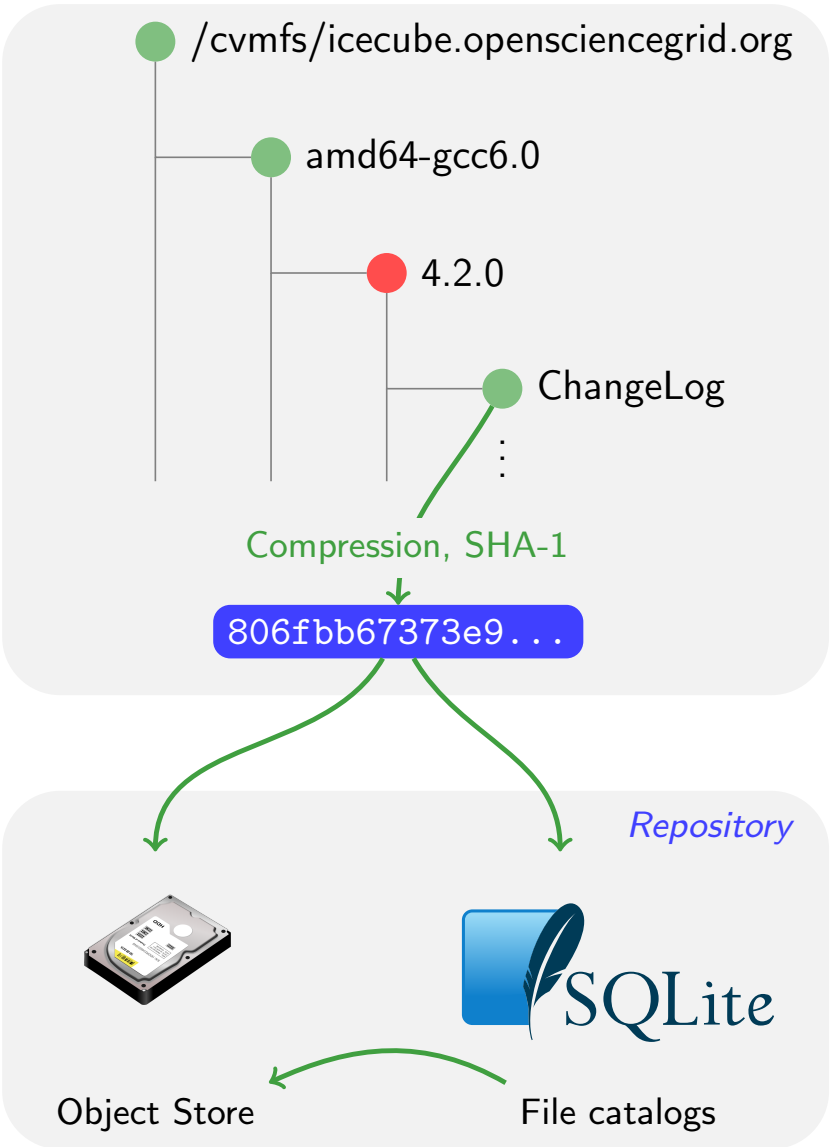
Backup Slides

Motivation: disk replication plus memory caches is not good enough for today's web services and data-intensive analysis applications

- Easily hundreds of RPCs to serve a request (e. g. a Facebook page),
  - very high cache hit rate necessary
  - unacceptable performance when cache is re-populated
- Keeping multiple replicas in memory is costly
- Nevertheless, we tend to program distributed storage like main memory



Content-Addressable Storage: Data Structures



Object Store

- Compressed files and chunks
- De-duplicated

File Catalog

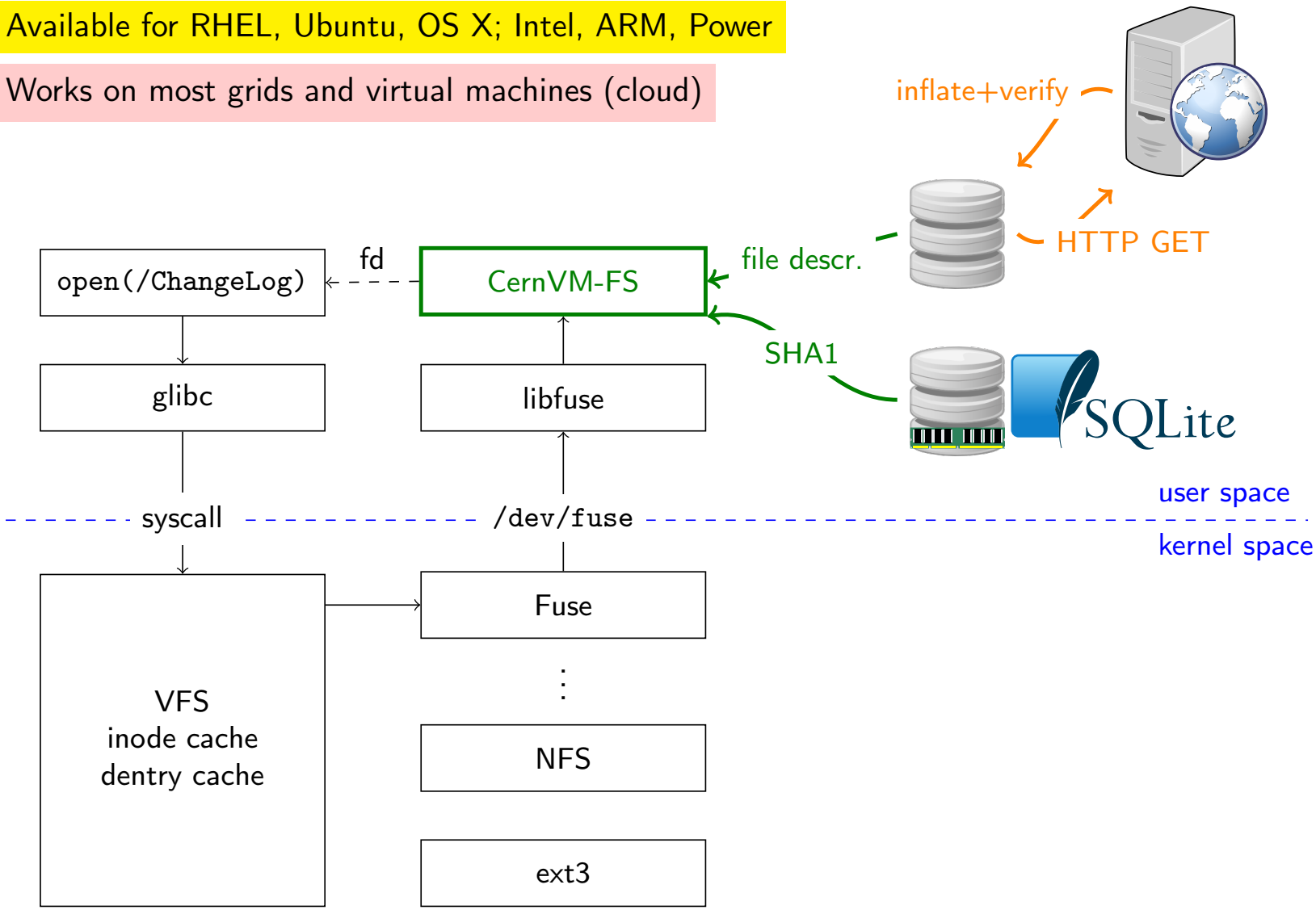
- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed
  $\Rightarrow$ integrity, authenticity
- Time to live
- Partitioned / Merkle hashes
  (possibility of sub catalogs)

$\Rightarrow$ **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