



# CernVM-FS - A Scalable and Low Maintenance Software Distribution Service

7th February 2013

## ① Introduction

## ② From Package Managers to a File System

## ③ Keeping the File System Client Benign in Heterogenous Environments

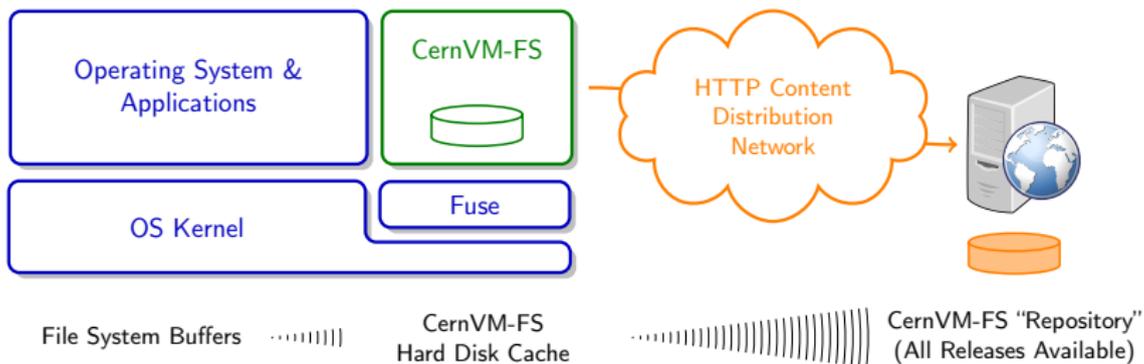
## ④ Use Case $\mu$ CernVM: Next Generation Virtual Machine

## ⑤ Summary



# CernVM File System at a Glance

Caching HTTP file system, optimized for software delivery



(Known) Users: ATLAS (+ Conditions Data), LHCb (+ Conditions Data), CMS, ALICE, NA61, NA49, BOSS, Geant4, AMS, LHC@Home 2.0

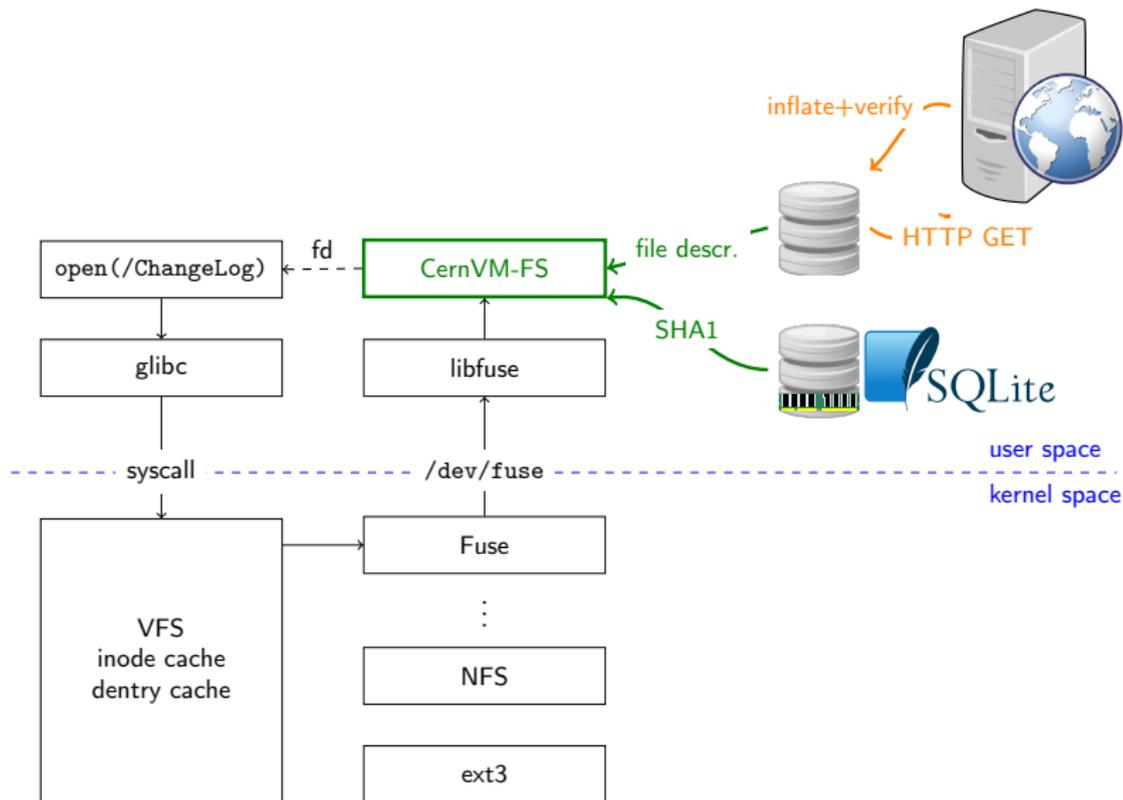
CDN: Full replicas at CERN, RAL, BNL, ASGC, FermiLab  
Site-local cache servers (Frontier Squids)

Avg. Load: Very modest,  
 $\approx 5$  MB/s, 20 requests per second on CERN Replica

Volume: 75 million objects (2010: 30 million), 5 TB (2010: 1 TB)



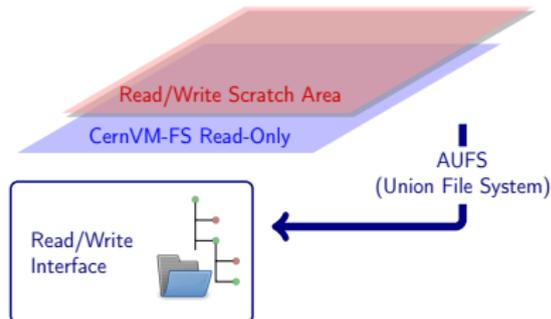
# The Client: A Read-Only File System in User Space





# The Server: A Publish Interface using Union-FS

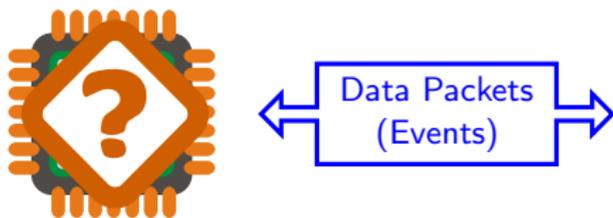
- Kernel-level Union File System AUFS
- < 5% performance loss (untar)



- 
- Fully POSIX-compliant read-write file system
  - Encapsulated change set in scratch area
  - In contrast to file-wise write: publishing of new snapshots

- ① Introduction
- ② From Package Managers to a File System
- ③ Keeping the File System Client Benign in Heterogenous Environments
- ④ Use Case  $\mu$ CernVM: Next Generation Virtual Machine
- ⑤ Summary







```
> cmsRun DiPhoton_Analysis_cfg.py
```





# Data Analysis Environment

```
> cmsRun DiPhoton_Analysis_cfg.py
```



Individual  
Analysis Code

*0.1 MLOC*

Experiment  
Software Framework

*4 MLOC*

High Energy Physics  
Libraries

*5 MLOC*

Compiler  
System Libraries  
OS Kernel

*20 MLOC*



# Data Analysis Environment

```
> cmsRun DiPhoton_Analysis_cfg.py
```



Individual  
Analysis Code

0.1 MLOC

Experiment  
Software Framework

4 MLOC

High Energy Physics  
Libraries

5 MLOC

Compiler  
System Libraries  
OS Kernel

20 MLOC

changing

stable

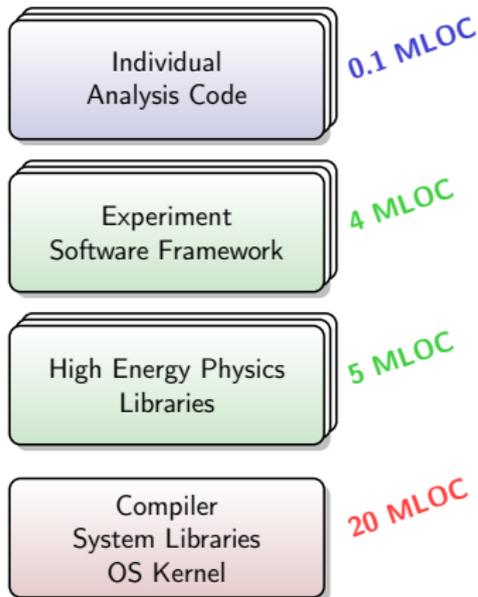


# Data Analysis Environment

```
> cmsRun DiPhoton_Analysis_cfg.py
```



Data Packets  
(Events)



changing  
stable

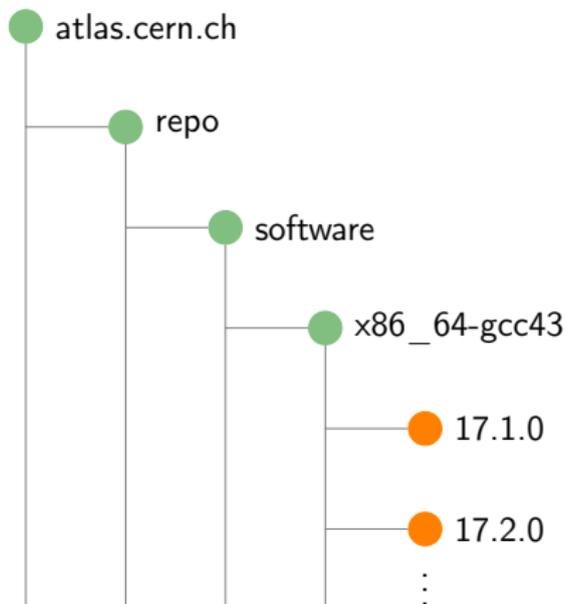
## Amplifying

- Frequent Updates
- Reproducibility
- Not easily chunkable



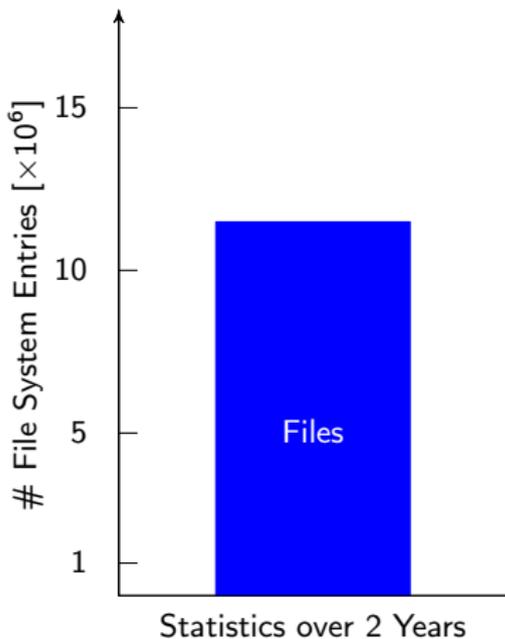
# Experiment Software from a File System Viewpoint

## Software Directory Tree

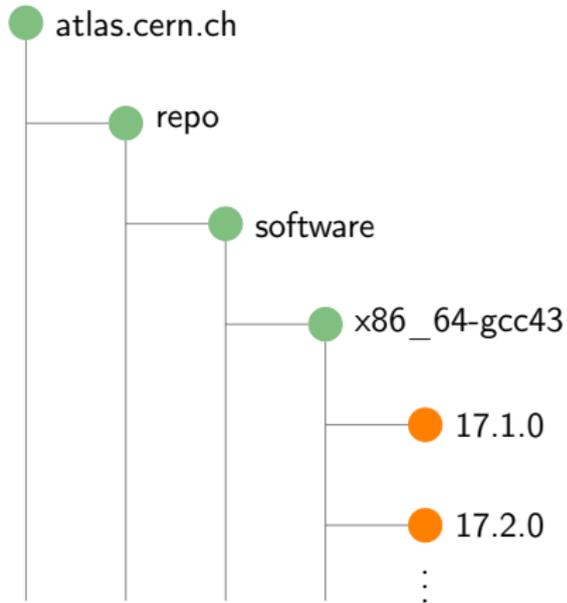




# Experiment Software from a File System Viewpoint

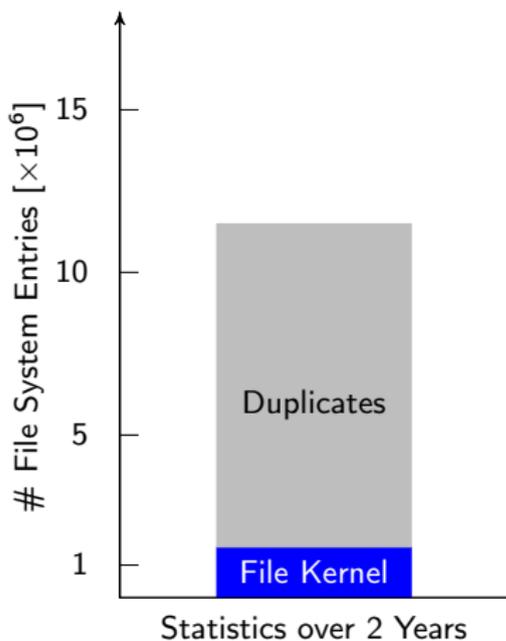


## Software Directory Tree

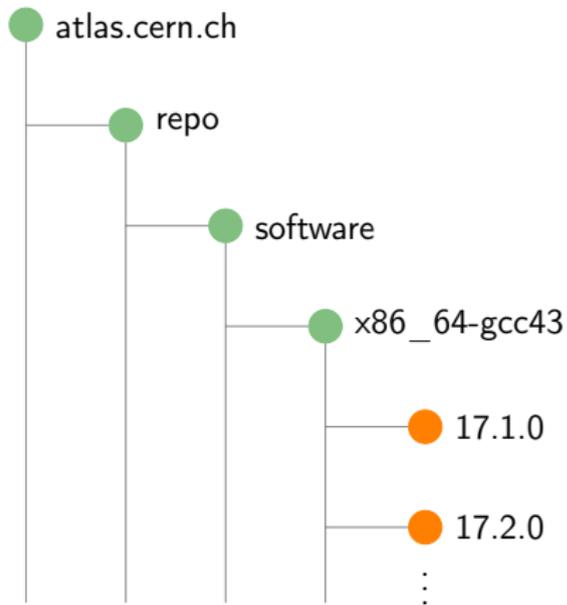




# Experiment Software from a File System Viewpoint

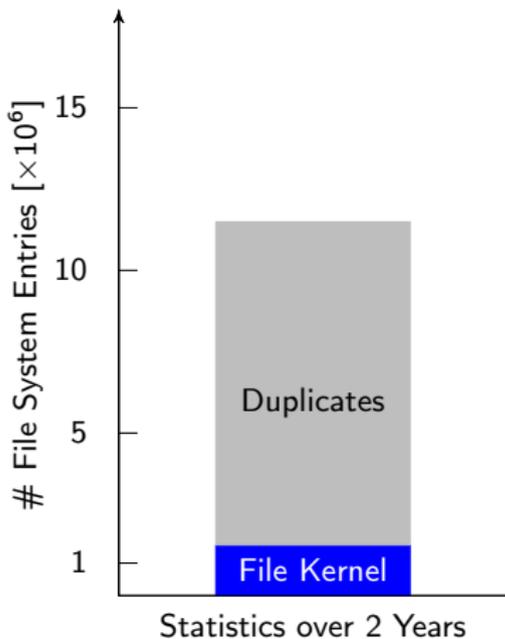


## Software Directory Tree

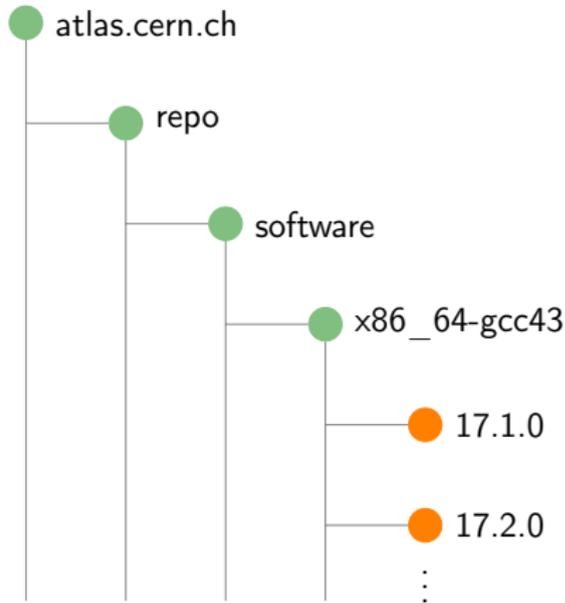




# Experiment Software from a File System Viewpoint



## Software Directory Tree

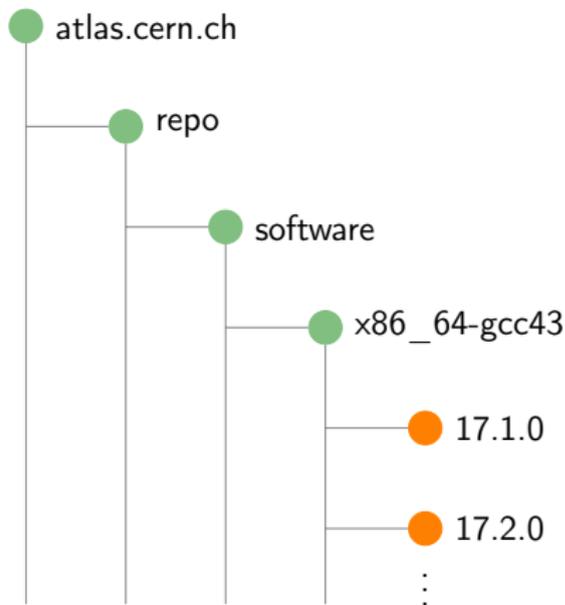
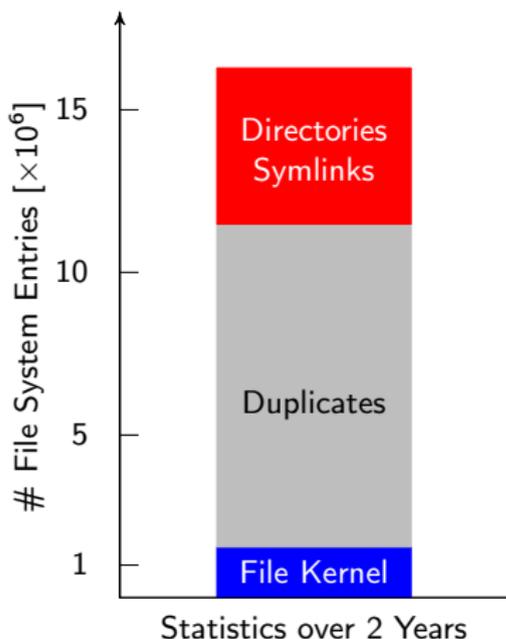


Between consecutive software versions: only  $\approx 15\%$  new files



# Experiment Software from a File System Viewpoint

## Software Directory Tree



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

Between consecutive software versions: only  $\approx 15\%$  new files



## Working Set as seen with CernVM

- $\approx 10\%$  of all available files are requested at runtime
- Median of file sizes:  $< 4$  kB

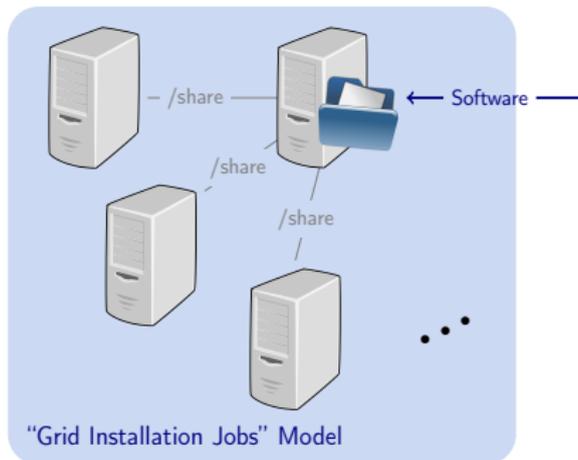


## Working Set as seen with CernVM

- $\approx 10\%$  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



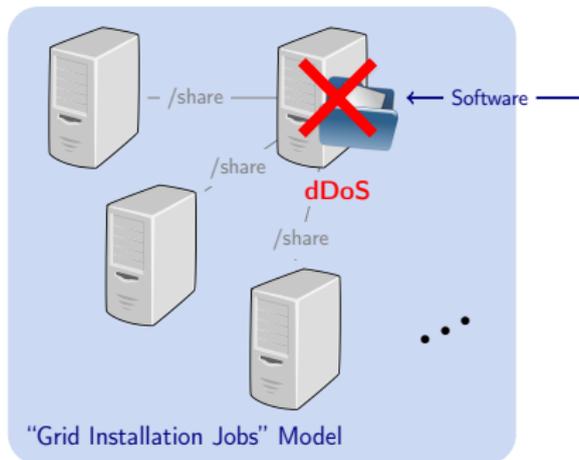


## Working Set as seen with CernVM

- $\approx 10\%$  of all available files are requested at runtime
- Median of file sizes:  $< 4$  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

<b>Software</b>	<b>Data</b>
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
Low latency	High throughput
Absolute paths	Any mount point
Open source	Confidential
WORM ("write-once-read-many")	
Versioned	



Based on ATLAS Figures 2012

## CernVM-FS

Global  
File System

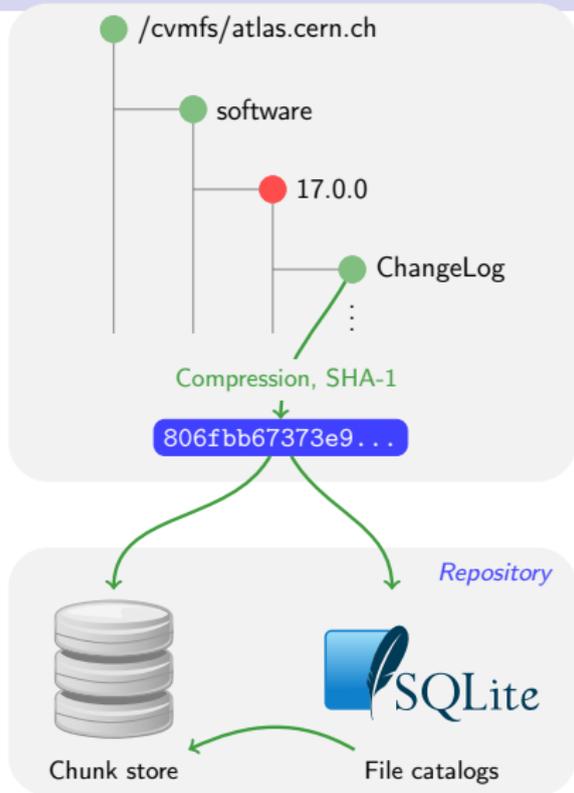
Content-Addressable  
Storage

Distributed  
Caching

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
Variant symlinks	No Symlinks
Absolute paths	Any mountpoint
Open source	Confidential
WORM (“write-once-read-many”) Versioned	



# Content-Addressable Storage in CernVM-FS



## Data Store

- Compressed chunks (files)
- Eliminates duplicates

## File Catalog

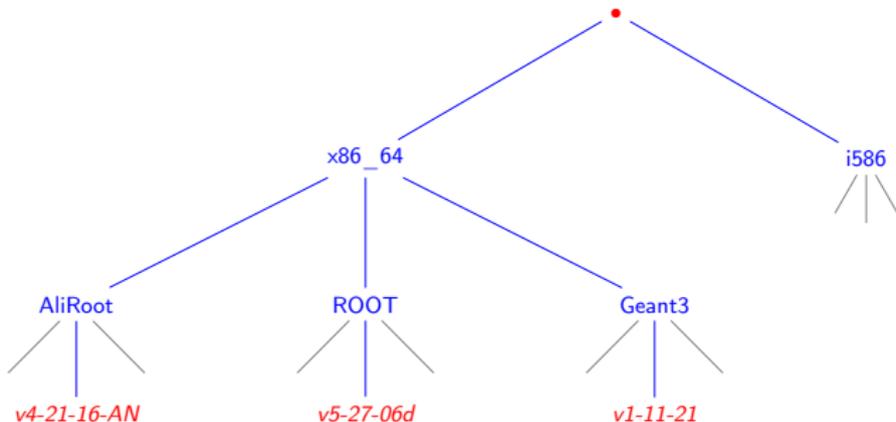
- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed  
⇒ integrity, authenticity
- Time to live
- Partitioned / Merkle hashes  
(user assisted)

⇒ Immutable files, trivial to check for corruption, consistency by snapshots  
≈ 6× reduction in number of files / volume



## Automatic Approaches

File based (Tolia et al. 2004), directory based (Kutzner 2008),  
global (Compostella et al. 2010)

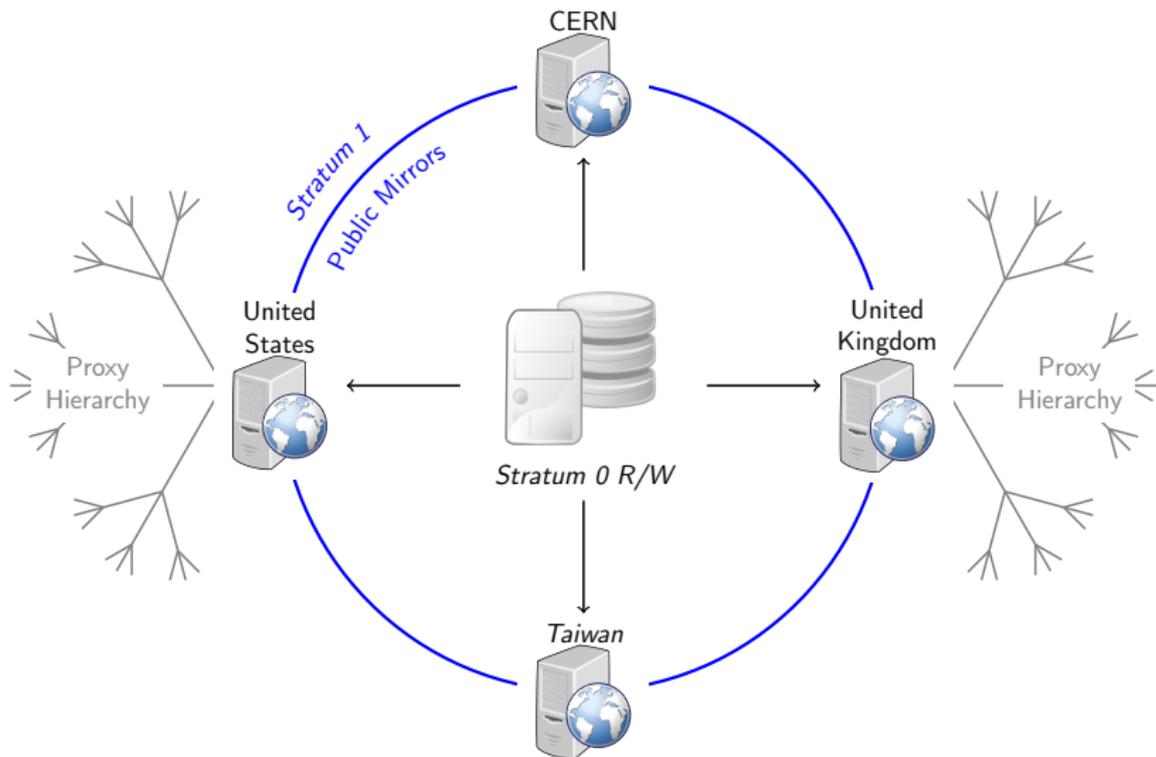


CernVM-FS: Semi-automatic, use of human knowledge about

- locality by software version
- locality by frequency of changes



# CernVM-FS Content Distribution

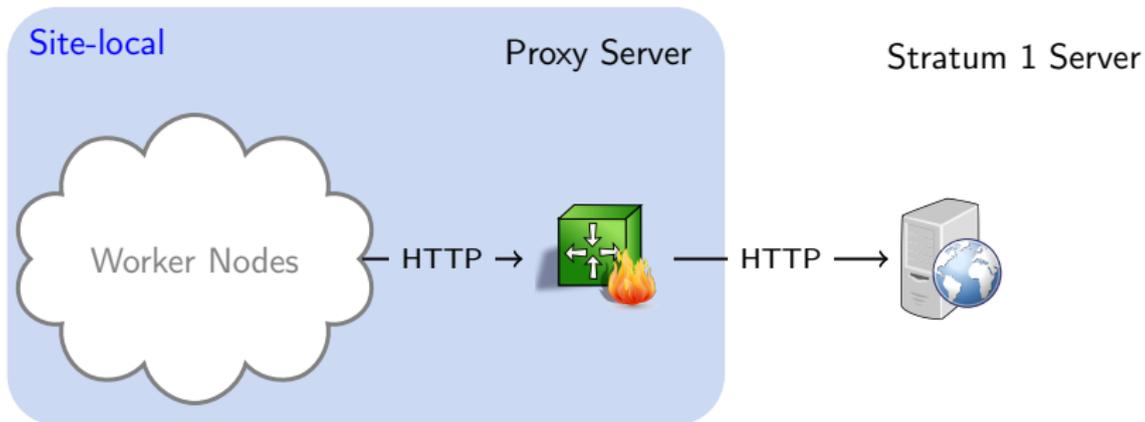


- ① Introduction
- ② From Package Managers to a File System
- ③ Keeping the File System Client Benign in Heterogenous Environments**
- ④ Use Case  $\mu$ CernVM: Next Generation Virtual Machine
- ⑤ Summary



# High-Availability by Horizontal Scaling

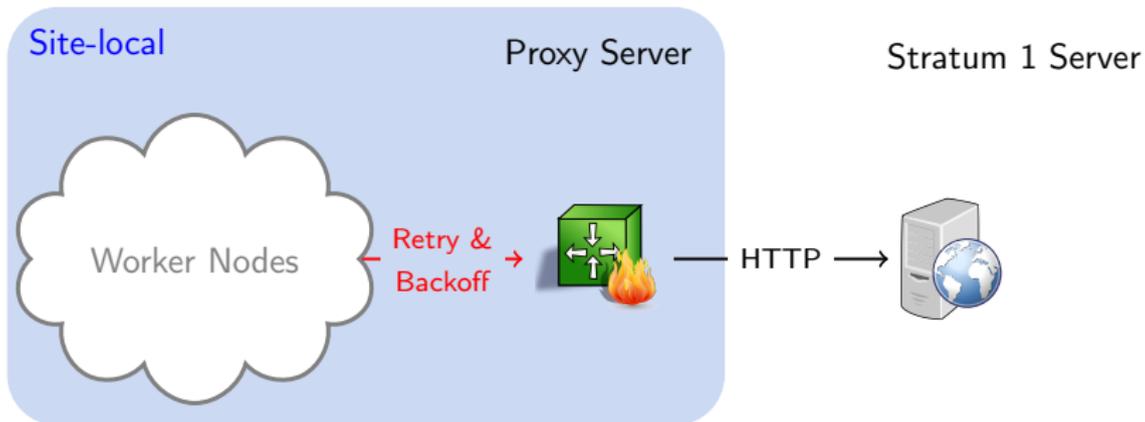
## Server side: stateless services





# High-Availability by Horizontal Scaling

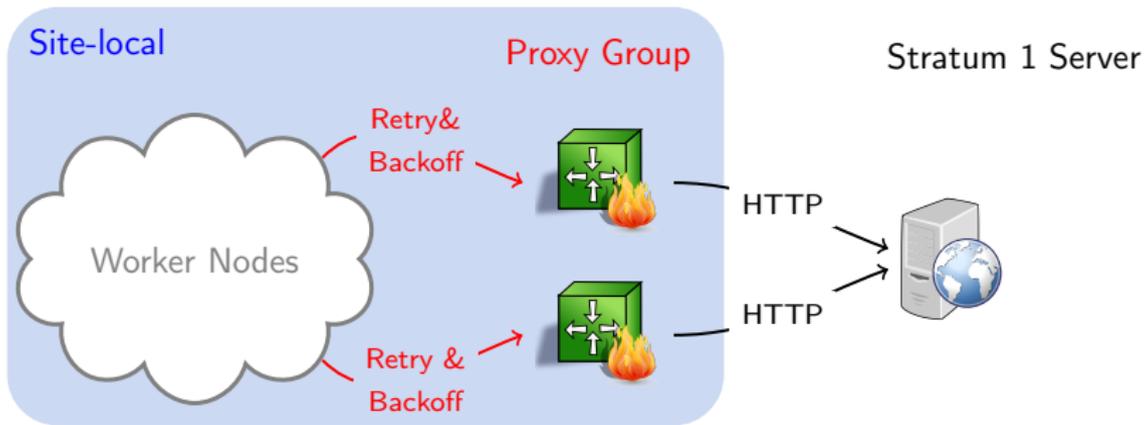
## Server side: stateless services





# High-Availability by Horizontal Scaling

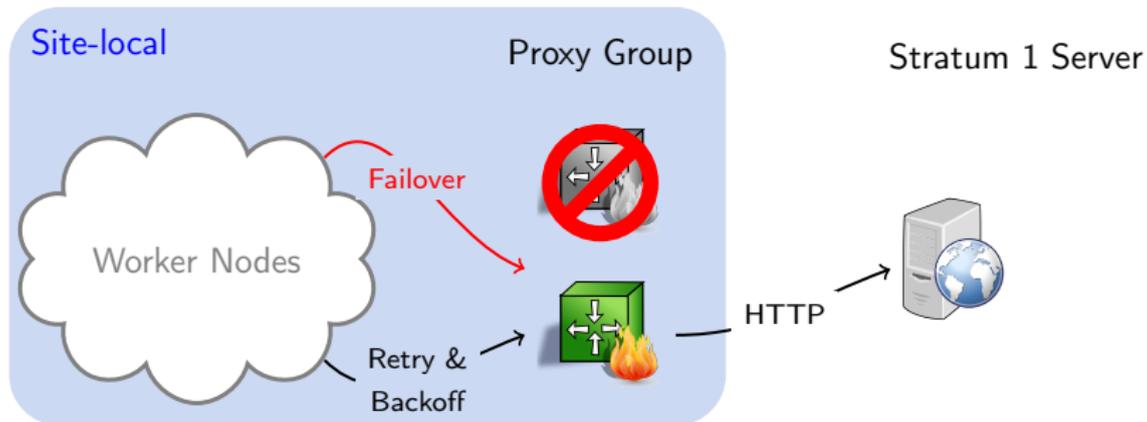
## Server side: stateless services





# High-Availability by Horizontal Scaling

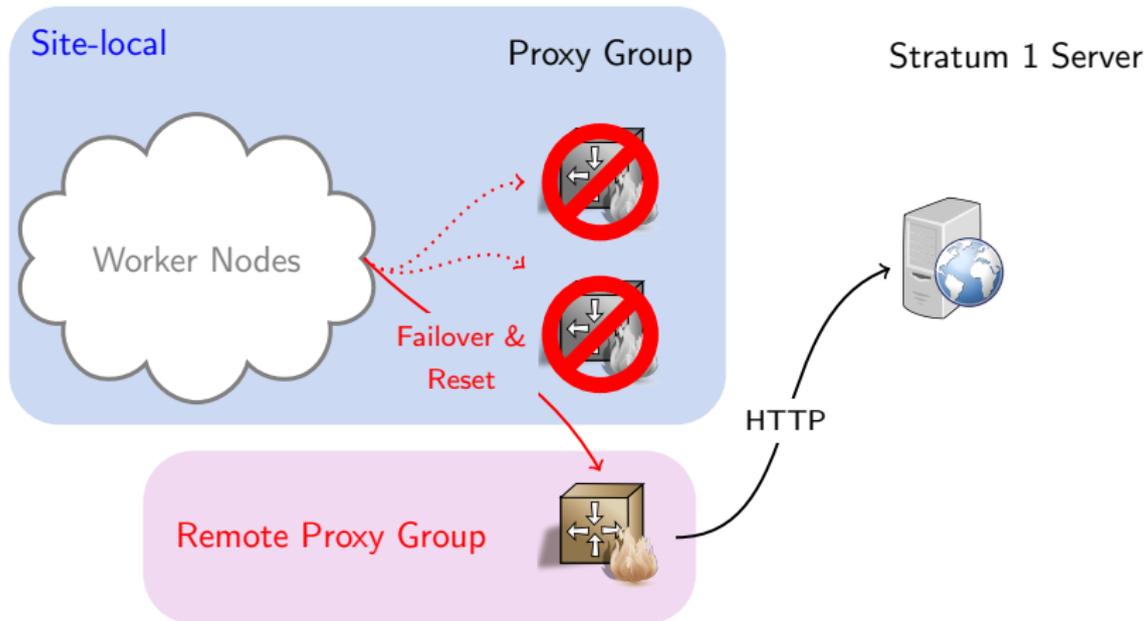
## Server side: stateless services





# High-Availability by Horizontal Scaling

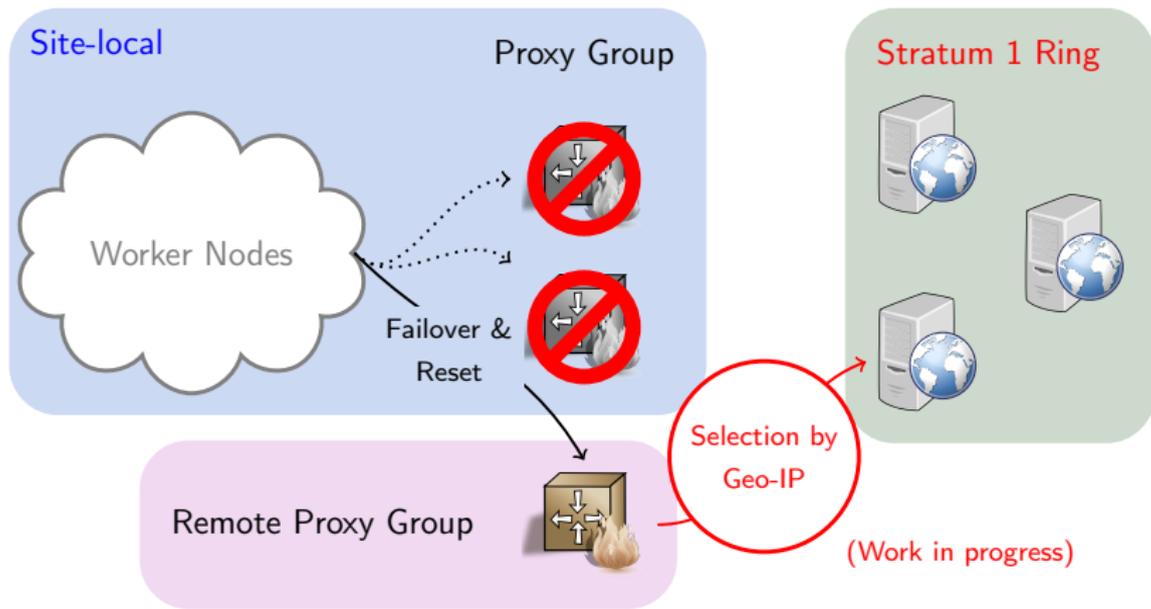
## Server side: stateless services





# High-Availability by Horizontal Scaling

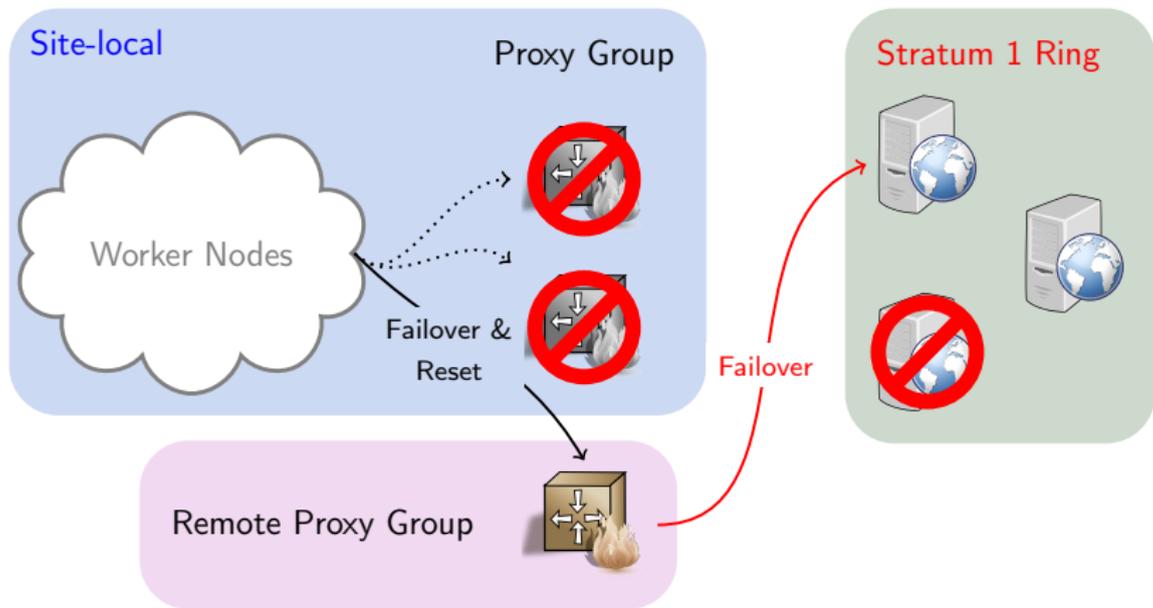
## Server side: stateless services





# High-Availability by Horizontal Scaling

## Server side: stateless services





# CernVM-FS Client in Heterogeneous Environments

In order to fully benefit from CernVM-FS, the file system has to be available on **all relevant computing resources**.

## Range of Environments:

Scientific Linux, Fedora, Ubuntu, SuSE, OS X  
1 core to 48+ cores  
Tens of mounted repositories  
Possibly no Fuse, disk-less server farms

## Portability:

- Portable C++ / POSIX code
- Library interface, connector to *Parrot* (by Dan Bradley)

## Scalability:

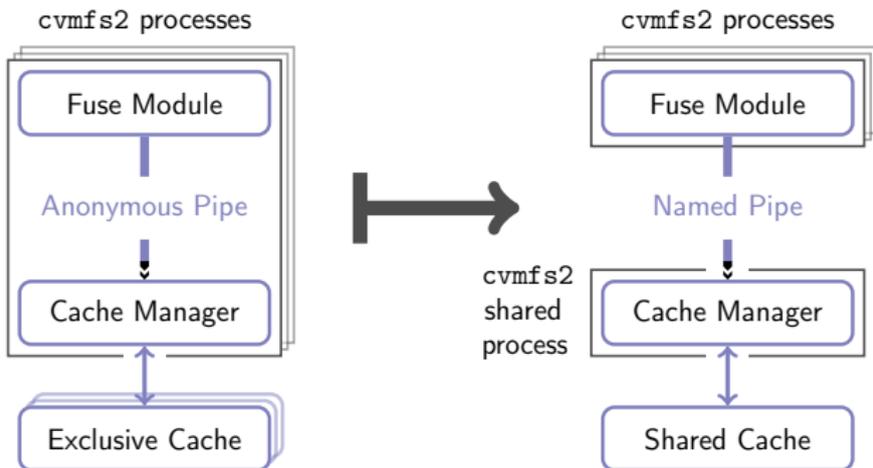
- Memory fragmentation  
open hash collision resolution  $\mapsto$   
linear probing  
path strings stored on the stack
- Concurrent file system access  
Fine-grained locking  
Asynchronous, parallel HTTP I/O
- Cache sharable among repositories
- **Hotpatch** functionality for Fuse client



# Shared Local Hard Disk Cache

**Issue:** Enforce shared *quota*, coordinated bookkeeping required

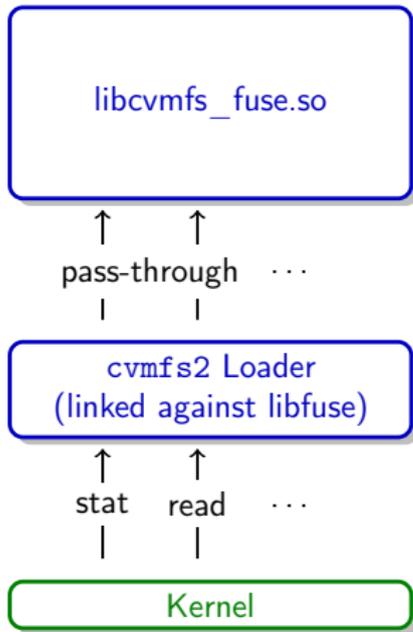
**Idea:** Turn the cache manager thread into a shared process



- No extra service: automatically spawned by first cvmfs mount point, automatically terminated by last unmount
- Named pipe can be turned into a network socket:  
**Foundation for distributed shared memory cache**



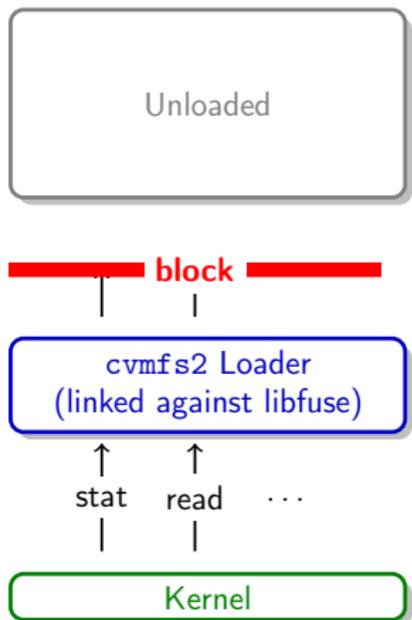
Addresses the issue of “**worker node draining**”  
when there is a new version of CernVM-FS



- A minimal loader implements the Fuse interface
- The logic is part of an (unloadable) shared library
- Very little state across file system calls: open files and open directories
- Can be also seen as a reload of parameters (like SIGHUP)



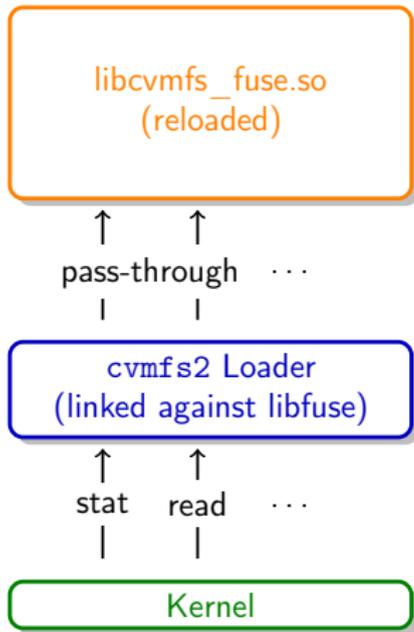
Addresses the issue of “**worker node draining**”  
when there is a new version of CernVM-FS



- A minimal loader implements the Fuse interface
- The logic is part of an (unloadable) shared library
- Very little state across file system calls: open files and open directories
- Can be also seen as a reload of parameters (like SIGHUP)



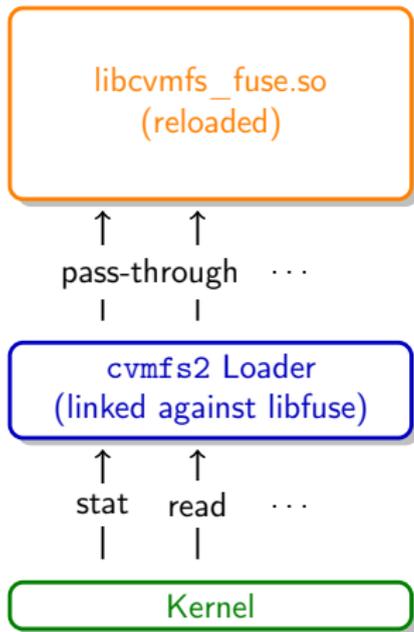
Addresses the issue of “**worker node draining**”  
when there is a new version of CernVM-FS



- A minimal loader implements the Fuse interface
- The logic is part of an (unloadable) shared library
- Very little state across file system calls: open files and open directories
- Can be also seen as a reload of parameters (like SIGHUP)



Addresses the issue of “**worker node draining**”  
when there is a new version of CernVM-FS



- A minimal loader implements the Fuse interface
- The logic is part of an (unloadable) shared library
- Very little state across file system calls: open files and open directories
- Can be also seen as a reload of parameters (like SIGHUP)

**(Just released)**



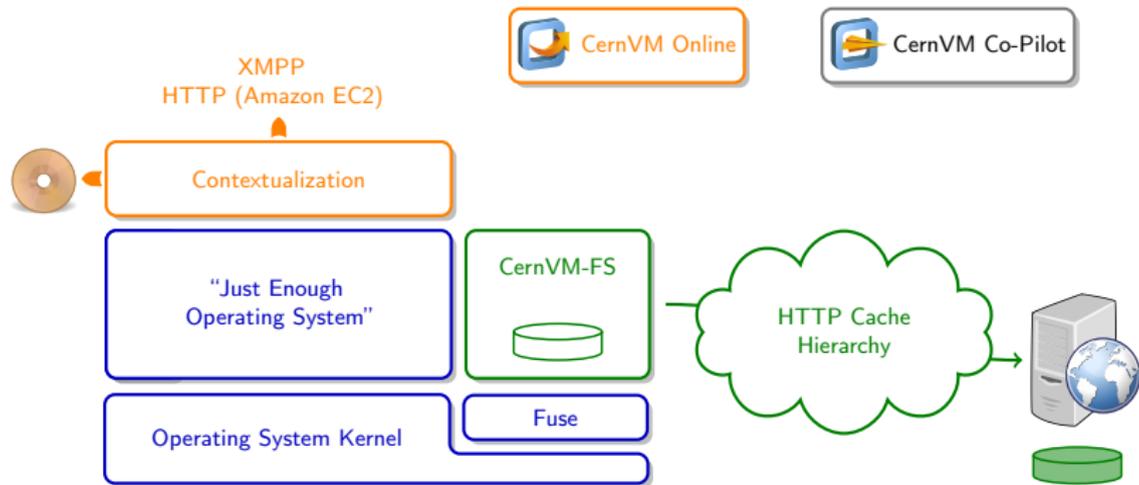
- ① As part of CernVM
  - Auto-configured
- ② As Fuse module on the worker node / workstation
  - Almost all of ATLAS, LHCb using it this way
  - Very little problems
  - “Private mounts” as normal user possible
- ③ NFS-exported Fuse module
  - Brings back infamous NFS bottleneck
  - Only solution for disk-less farms
  - DESY: 2 k nodes on CernVM-FS / NFS
- ④ As part of the Grid job
  - Using library interface + Parrot connector
  - Runs as normal user

**Squids:** Site-local, Frontier, Public Service

- ① Introduction
- ② From Package Managers to a File System
- ③ Keeping the File System Client Benign in Heterogenous Environments
- ④ Use Case  $\mu$ CernVM: Next Generation Virtual Machine
- ⑤ Summary



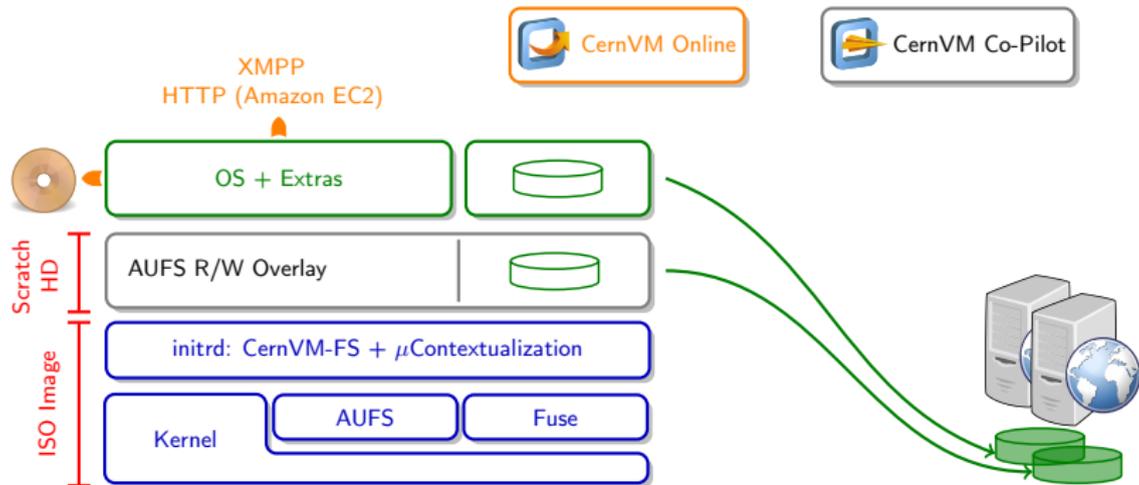
## Classic CernVM



- Uniform and portable environment for physics data processing
- Minimal operating system derived from application dependencies
- Easy to maintain and to distribute



## $\mu$ CernVM



**Idea:** Operating system on CernVM-FS

Instead of 400 MB hard disk image: 10 MB ISO image + 100 MB cache.

- *Not* a LiveCD, *not* a diskless node
- ⇒ Operating System on Demand



# Booting $\mu$ CernVM

Work in progress

```
MicroCernVM
ISOLINUX 4.06 2012-10-23 ETCD Copyright (C) 1994-2012 H. Peter Anvin et al
early console in decompress_kernel

Decompressing Linux... Parsing ELF... done.
Booting the kernel.
[ 1.277254] acpiphp_ibm: ibm_acpiphp_init: acpi_walk_namespace failed
[ 1.459769] sd 2:0:0:0: [sda] Assuming drive cache: write through
[ 1.460199] sd 2:0:0:0: [sda] Assuming drive cache: write through
[ 1.461662] sd 2:0:0:0: [sda] Assuming drive cache: write through

* Welcome to micro-CernUM
* Beta release 1.2

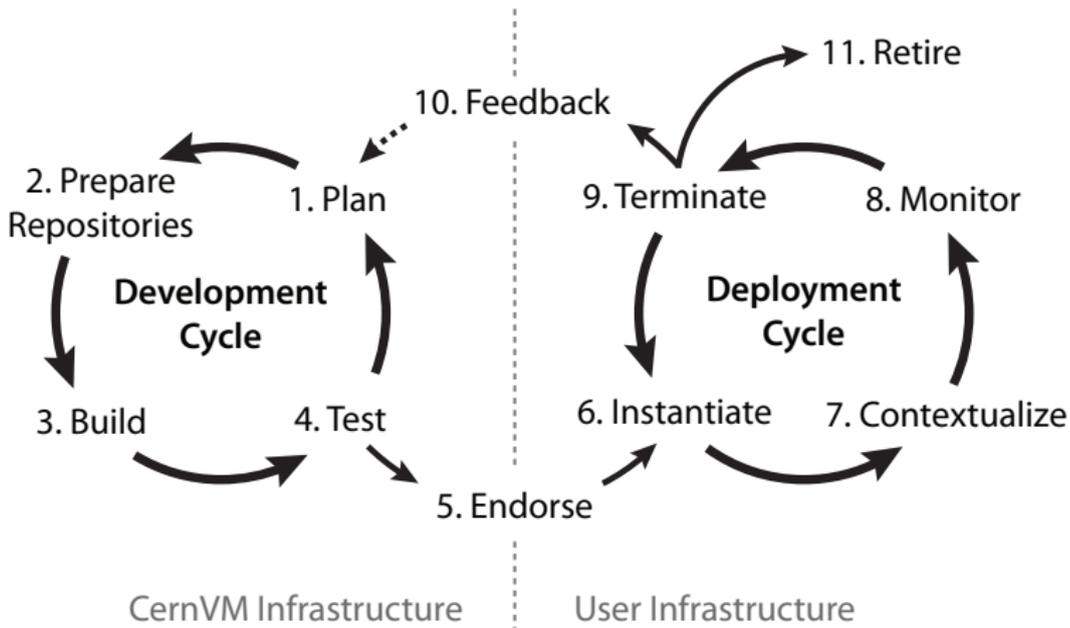
[INF] Setting up environment... check
[INF] Loading predefined modules... check
[INF] Starting networking... check
[INF] Mounting root filesystem... check
[INF] Starting CernUM File System... check

mount: mount point /proc/bus/usb does not exist
Welcome to Scientific Linux
Starting udev: _
```



# $\mu$ CernVM Changes the VM Life Cycle

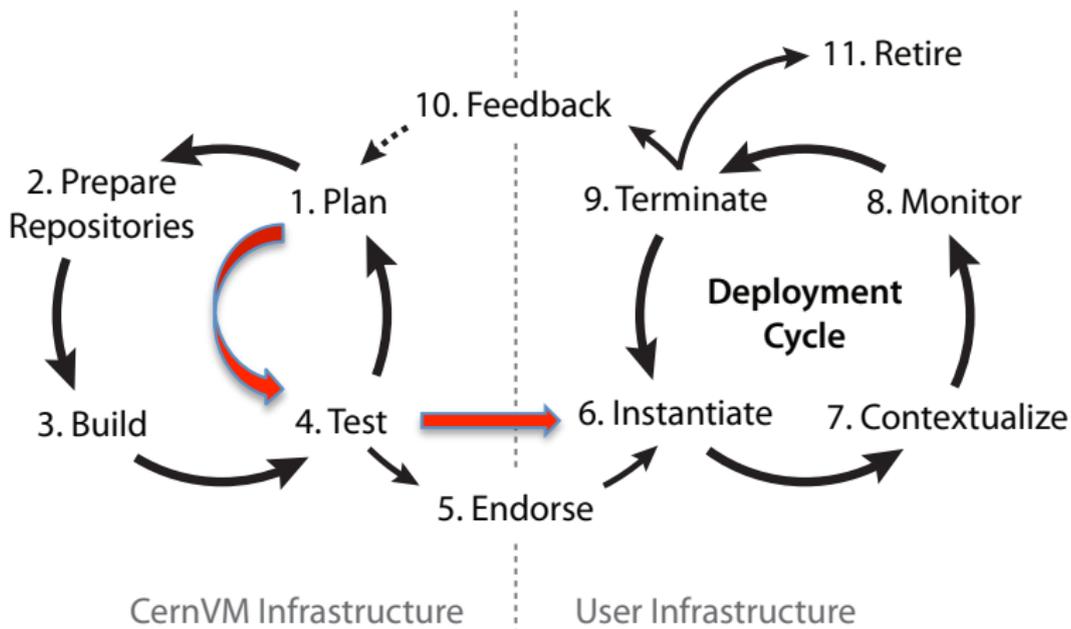
Work in progress





# $\mu$ CernVM Changes the VM Life Cycle

Work in progress



Avoids: Image Building

Solves: Image Distribution

Options for updating: **stay, diverge, rebase**

CernVM-FS snapshots facilitate **long-term data preservation**

- ① Introduction
- ② From Package Managers to a File System
- ③ Keeping the File System Client Benign in Heterogenous Environments
- ④ Use Case  $\mu$ CernVM: Next Generation Virtual Machine
- ⑤ Summary



## CernVM-FS

- Publish-subscribe file system
- Based on snapshots, distributed caching, content-addressable storage
- Very low network traffic
- Optimized for small files, heavy meta-data operation

## Use cases

- Software Distribution
- Distribution of Conditions Data
- Operating System for the Cloud
- Building block of long-term data preservation

---

Source code: <https://github.com/cvmfs/cvmfs>

Downloads: <http://cernvm.cern.ch/portal/filesystem/downloads>

Nightly builds: <https://ecsft.cern.ch/dist/cvmfs>

Mailing lists: [cvmfs-talk@cern.ch](mailto:cvmfs-talk@cern.ch), [cvmfs-devel@cern.ch](mailto:cvmfs-devel@cern.ch)

## ⑥ Backup Slides



## 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

## Mount helpers

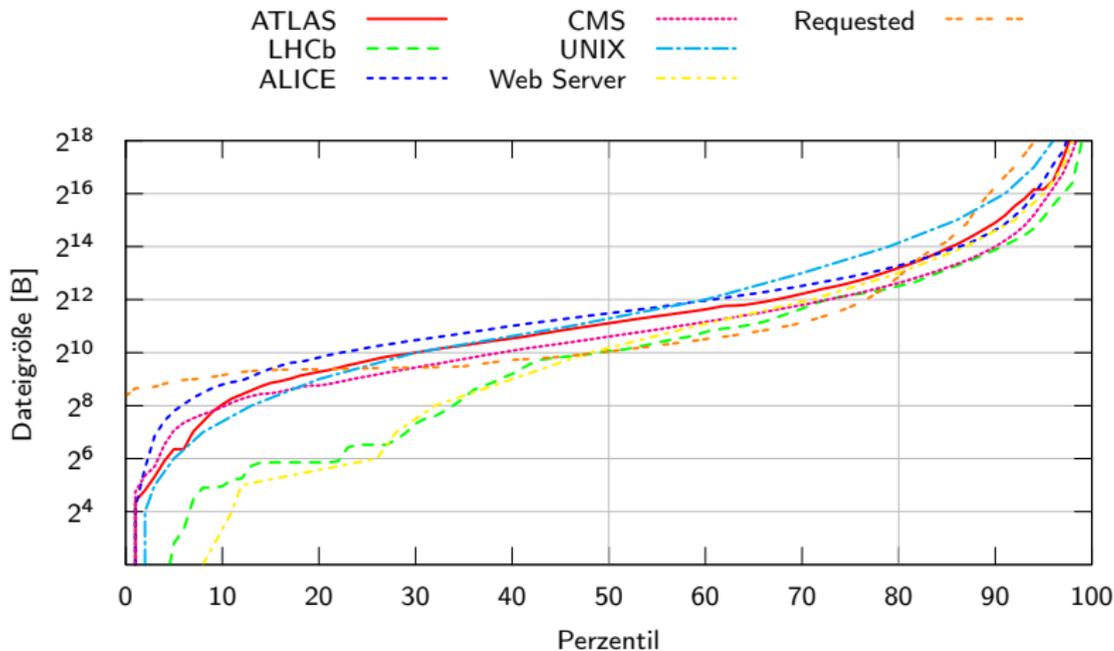
- Setup environment (number of file descriptors, access rights, ...)
- Used by autofs on /cvmfs
- Used by /etc/fstab or mount as root  
mount -t cvmfs atlas.cern.ch  
/cvmfs/atlas.cern.ch

## Diagnostics

- Nagios check available
- cvmfs\_config probe
- cvmfs\_config chksetup
- cvmfs\_fsck
- cvmfs\_talk, connect to running instance



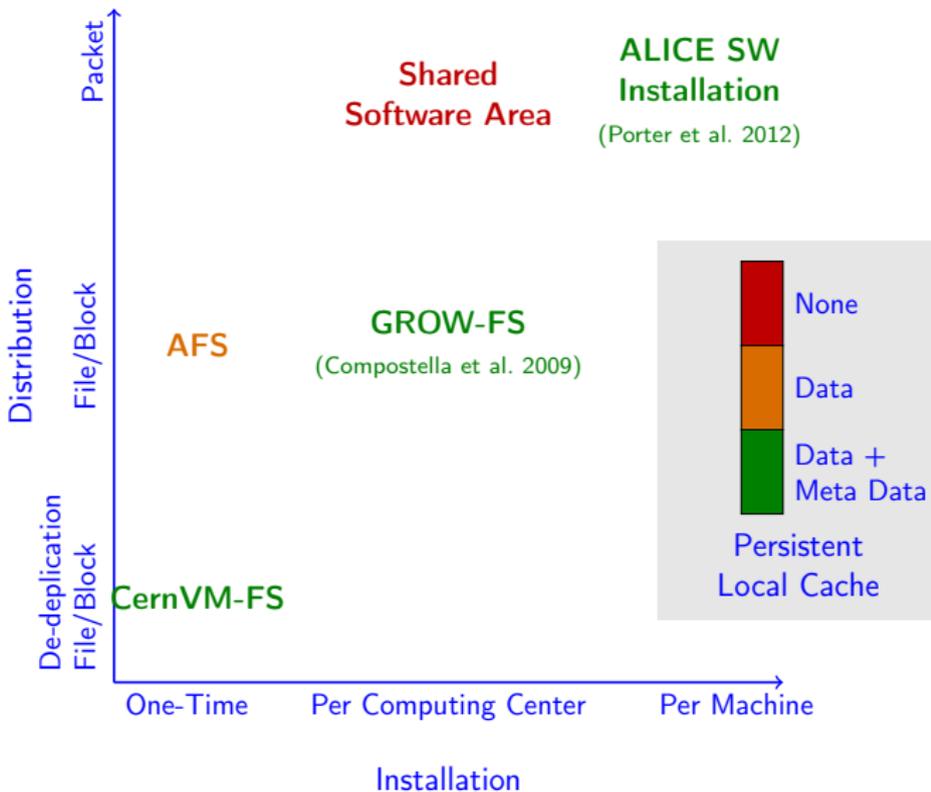
# Cumulative File Size Distribution



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



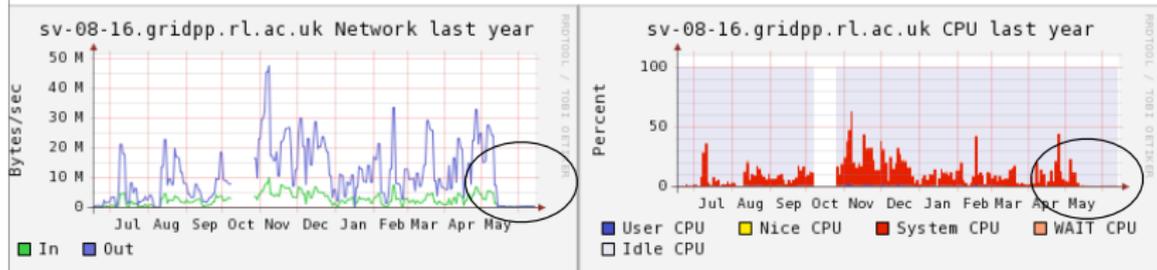
# Software Distribution Systems in HEP



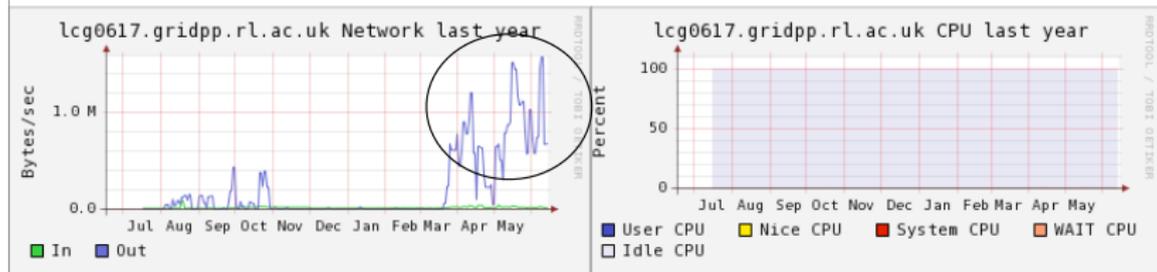


# Comparison CernVM-FS / NFS I

## •NFS Atlas SW Server Loads - switched Atlas to CVMFS in May



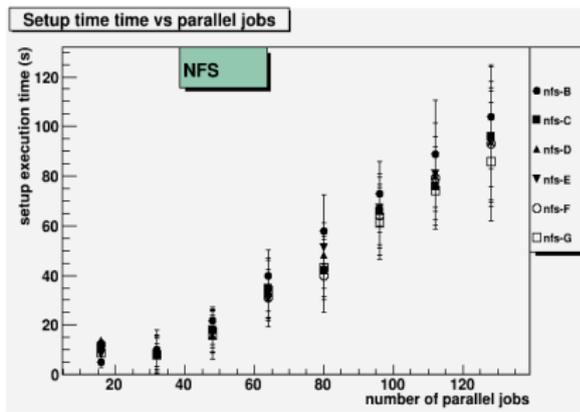
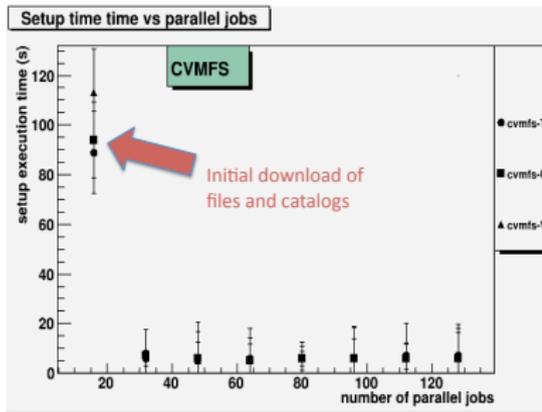
## •Site (cluster) Squid Server loads - this is just one of the two



(Source: Collier 2011)



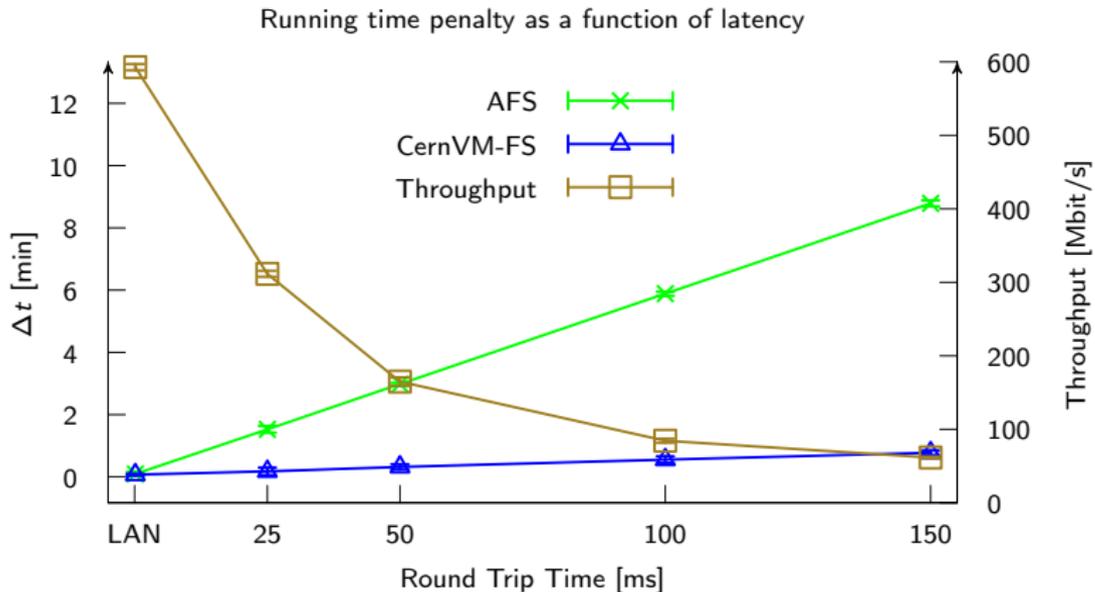
# Comparison CernVM-FS / NFS II



(Source: Lanciotti 2011)



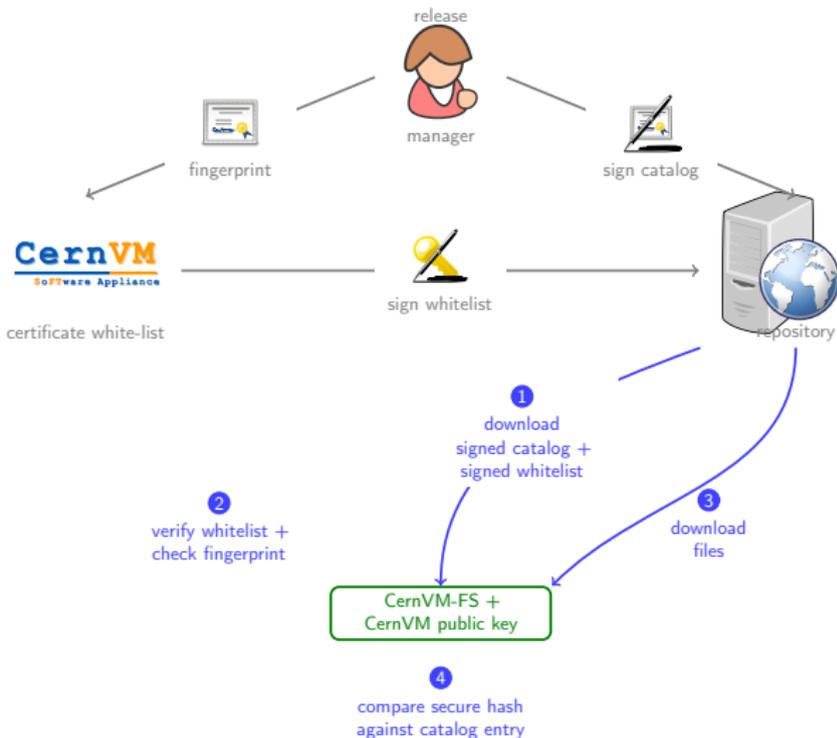
## Running the "StressHepix" benchmark





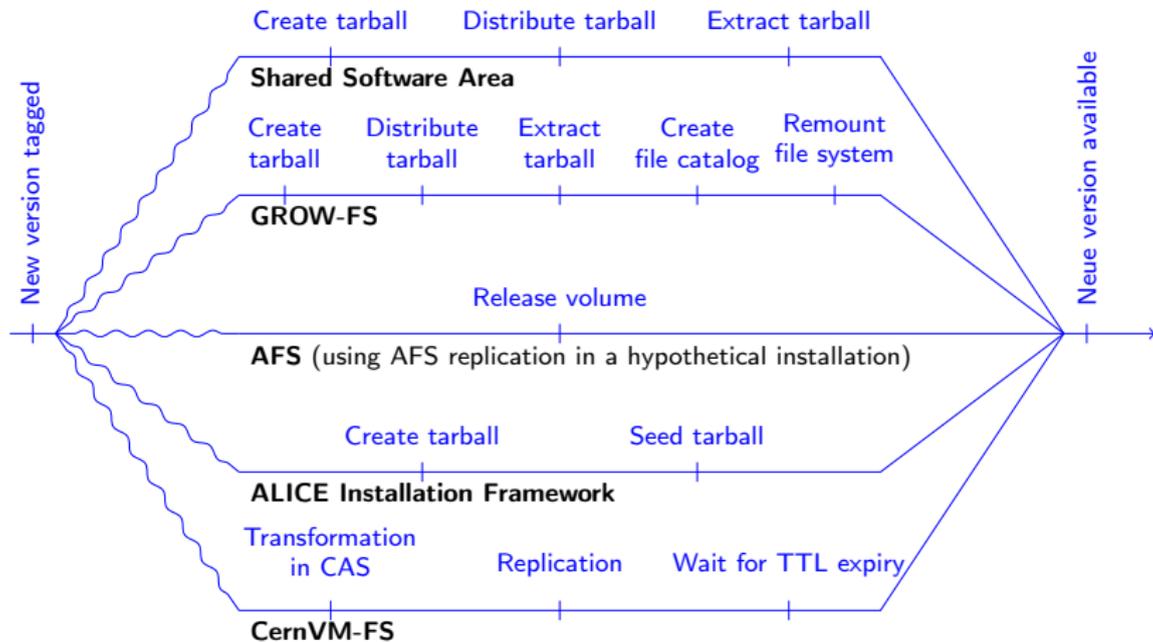
# Integrity and Authenticity

**Principle:** Digitally signed root hash and (white-)list of permitted signing certificates





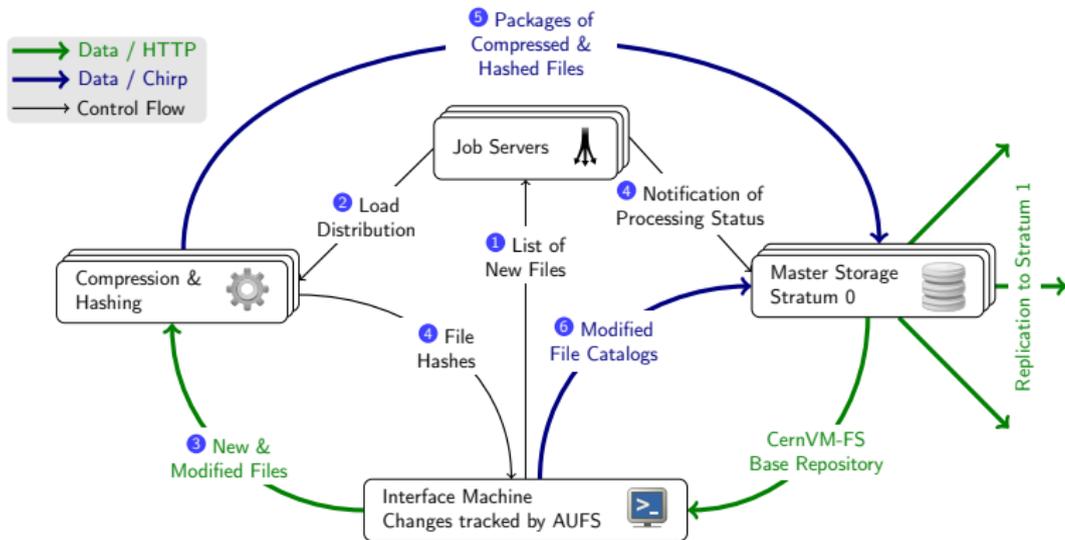
# Turn-Around for Software Distribution



**Improvement:** from days (Grid Installation Jobs) to hours



# CernVM-FS Distributed Storage Backend



**Roles:** File System Interface, Worker Node, Job Manager, Master Storage (+ Stratum 0 Webserver, Signing Server)

**Protocols:** Chirp, HTTP

**Storage Interface:** Put, Get, Rename/Commit (on Stratum 0)



- **Decentralized Data Storage and Processing in the Context of the LHC Experiments at CERN**  
*J. Blomer*, PhD Thesis, TU Munich, 2012
- **CernVM-FS: Delivering Scientific Software to Globally Distributed Computing Resources**  
*J. Blomer, P. Buncic, and T. Fuhrmann*  
Proc. of the 1st Workshop on Network-Aware Data Management held in conjunction with the IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), Seattle, 2011
- **A practical approach to virtualization in HEP**  
*P. Buncic, C. Aguado Sánchez, J. Blomer, A. Harutyunyan, and M. Mudrinic*  
The European Physical Journal Plus, **126**(1), 2011
- **Distributing LHC Application Software and Conditions Databases using the CernVM File System**  
*J. Blomer, C. Aguado Sánchez, P. Buncic, and A. Harutyunyan*  
Journal of Physics: Conference Series, **331**, 2011
- **A Fully Decentralized File System Cache for the CernVM-FS**  
*J. Blomer and T. Fuhrmann*  
Proc. 10th Computer and Communications Networks (ICCCN'10), Zürich, 2010
- **LHC Cloud Computing with CernVM**  
*B. Segal, P. Buncic, D. Garcia Quintas, C. Aguado Sánchez, J. Blomer, et al.*  
Proceedings of Science, **ACAT**(004), 2010