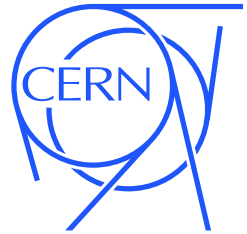


CernVM Blueprint for Long-term Data Preservation

Jakob Blomer

CERN PH-SFT

Base Technology: Virtual Machines

Virtual machines enable historic software environments
on today's infrastructure.

Base Technology: Virtual Machines

Virtual machines enable historic software environments
on today's infrastructure.

Add-On 1: CernVM File System

CernVM-FS is a *versioning* and *snapshotting* file system
used it to make the virtual machine's content accountable.

Overview

Base Technology: Virtual Machines

Virtual machines enable historic software environments
on today's infrastructure.

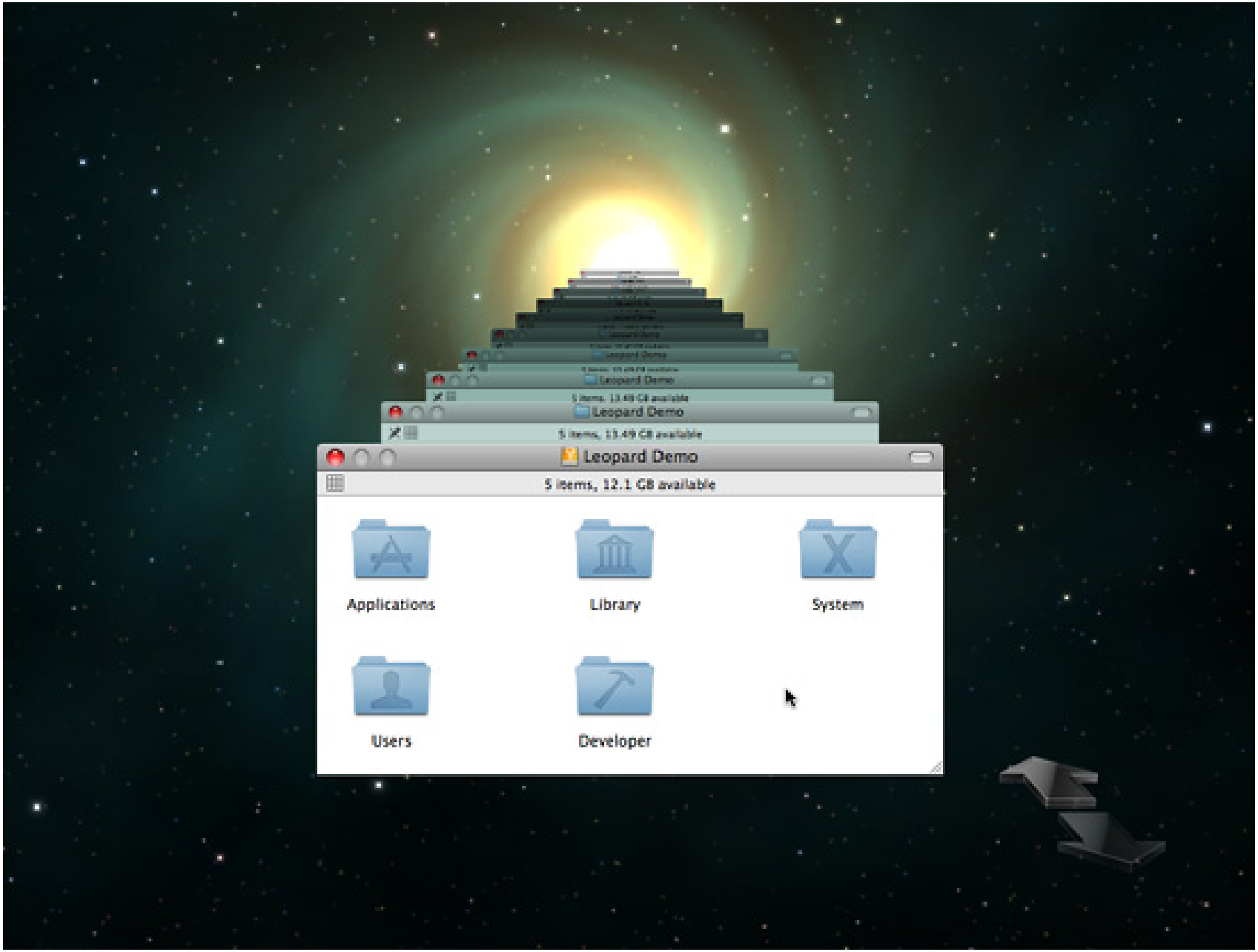
Add-On 1: CernVM File System

CernVM-FS is a *versioning* and *snapshotting* file system
used it to make the virtual machine's content accountable.

Add-On 2: CernVM Contextualization Agent

Supports textual specification for interacting CernVMs.
A *historic analysis cluster* is spawned from a single virtual machine image.

A Time Machine for the Analysis Environment



Rational of Preserving the Software Environment

1. Processing of legacy data
   - Software implicitly encodes knowledge about the correct interpretation of the data
   - After substantial upgrades and modifications of the detector, the new software might lose this legacy knowledge
   - After experiment decommission, porting and validation of software is likely to end

Rational of Preserving the Software Environment

1. Processing of legacy data
   - Software implicitly encodes knowledge about the correct interpretation of the data
   - After substantial upgrades and modifications of the detector, the new software might lose this legacy knowledge
   - After experiment decommission, porting and validation of software is likely to end
2. Validation of new software versions
   - Otherwise, if the new software can process legacy data, comparison with historic version provides input for validation

Rational of Preserving the Software Environment

1. Processing of legacy data
   - Software implicitly encodes knowledge about the correct interpretation of the data
   - After substantial upgrades and modifications of the detector, the new software might lose this legacy knowledge
   - After experiment decommission, porting and validation of software is likely to end
2. Validation of new software versions
   - Otherwise, if the new software can process legacy data, comparison with historic version provides input for validation

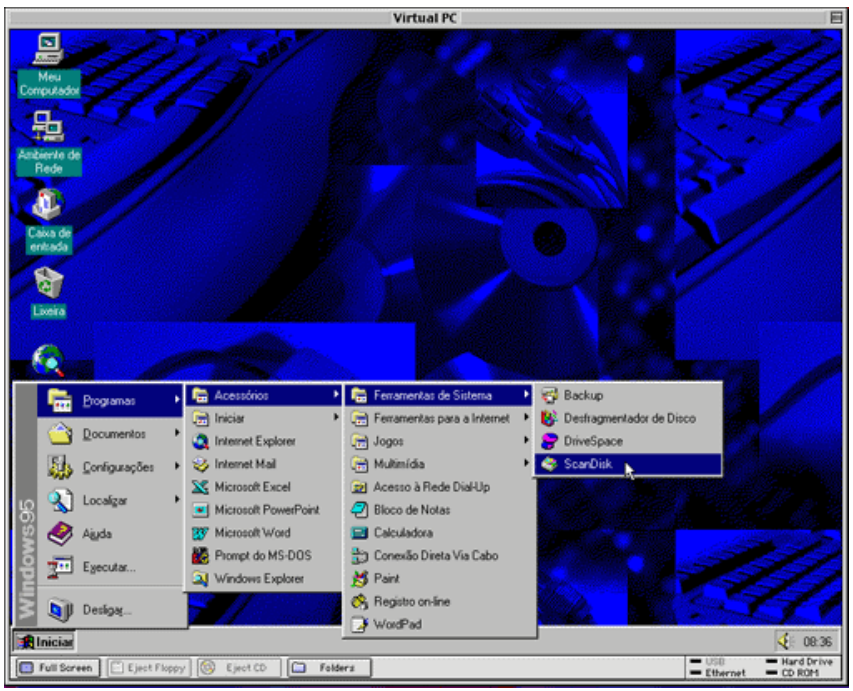
Alternative to preserving: eternal porting and validation

Potential of Virtualization Technology

. . . with reasonable performance

- Very efficient on Intel architecture
- Blessed by almost *30 years* backwards compatibility

Efficient virtualization across architectures:



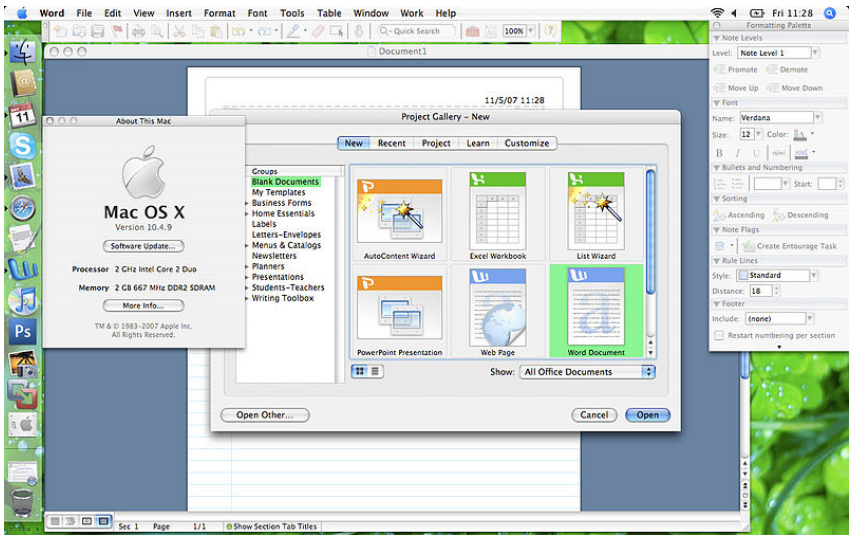
- Connectix Virtual PC ('90)
- Intel on PowerPC
- Windows, OS/2, Redhat Linux on Mac OS

Potential of Virtualization Technology

. . . with reasonable performance

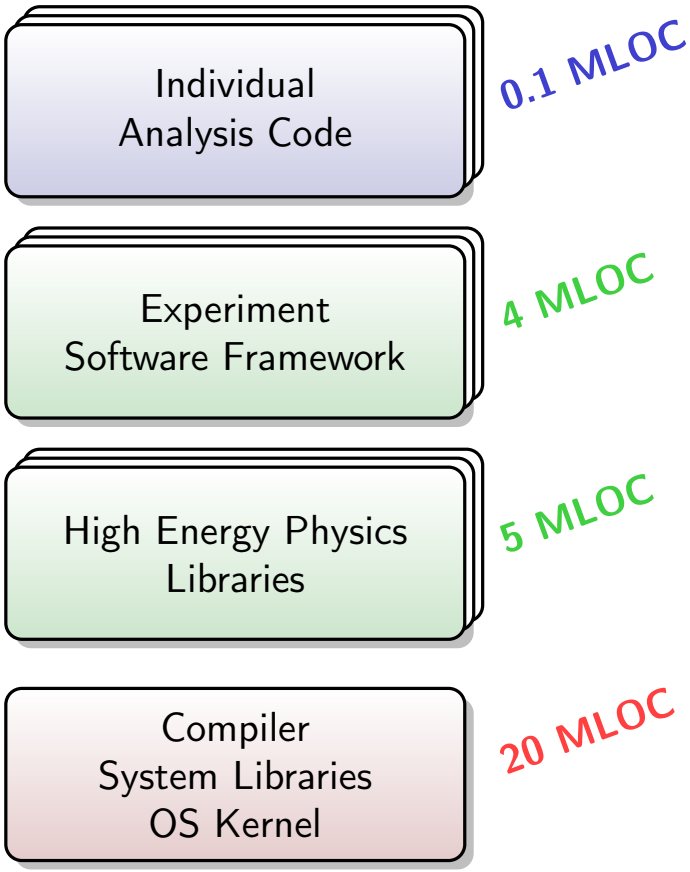
- Very efficient on Intel architecture
- Blessed by almost *30 years* backwards compatibility

Efficient virtualization across architectures:



- Apple Rosetta (2006)
- PowerPC on Intel
- Speed: ≈50% of latest PowerPC

HEP Data Processing Environment



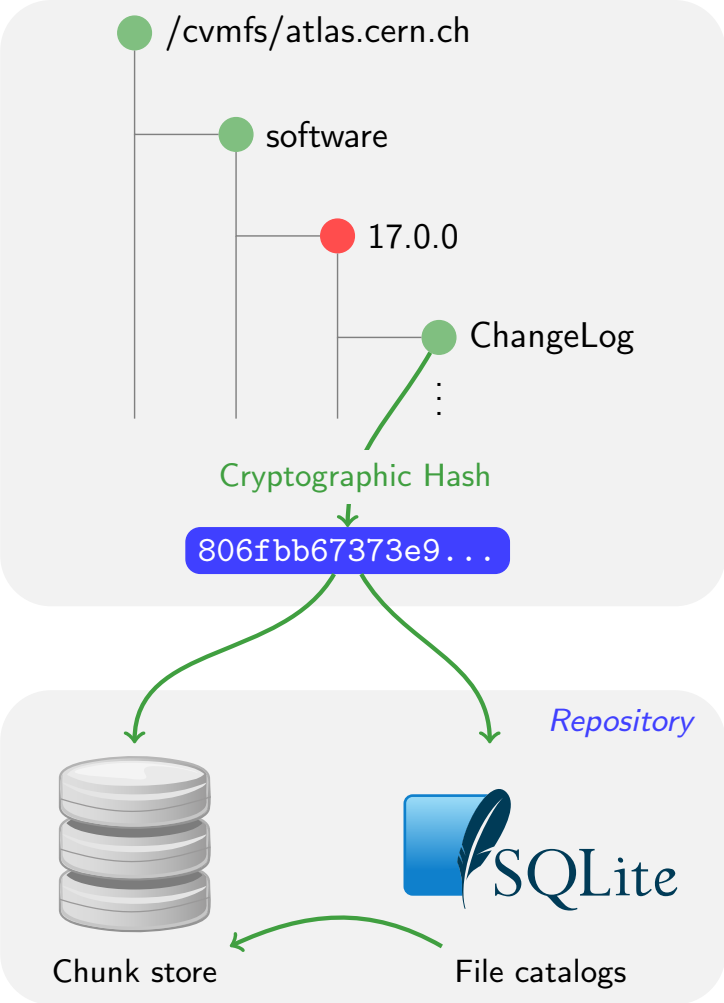
changing

stable

Amplifying

- Frequent Updates
- Not a single binary – a development environment
- Hundreds of libraries with partially untracked dependencies
- Not easily chunkable
- *Not easily packagable*

Versioning and Snapshots in CernVM-FS



Data Store

- Eliminates duplicates
- Never deletes, archiving

File Catalog

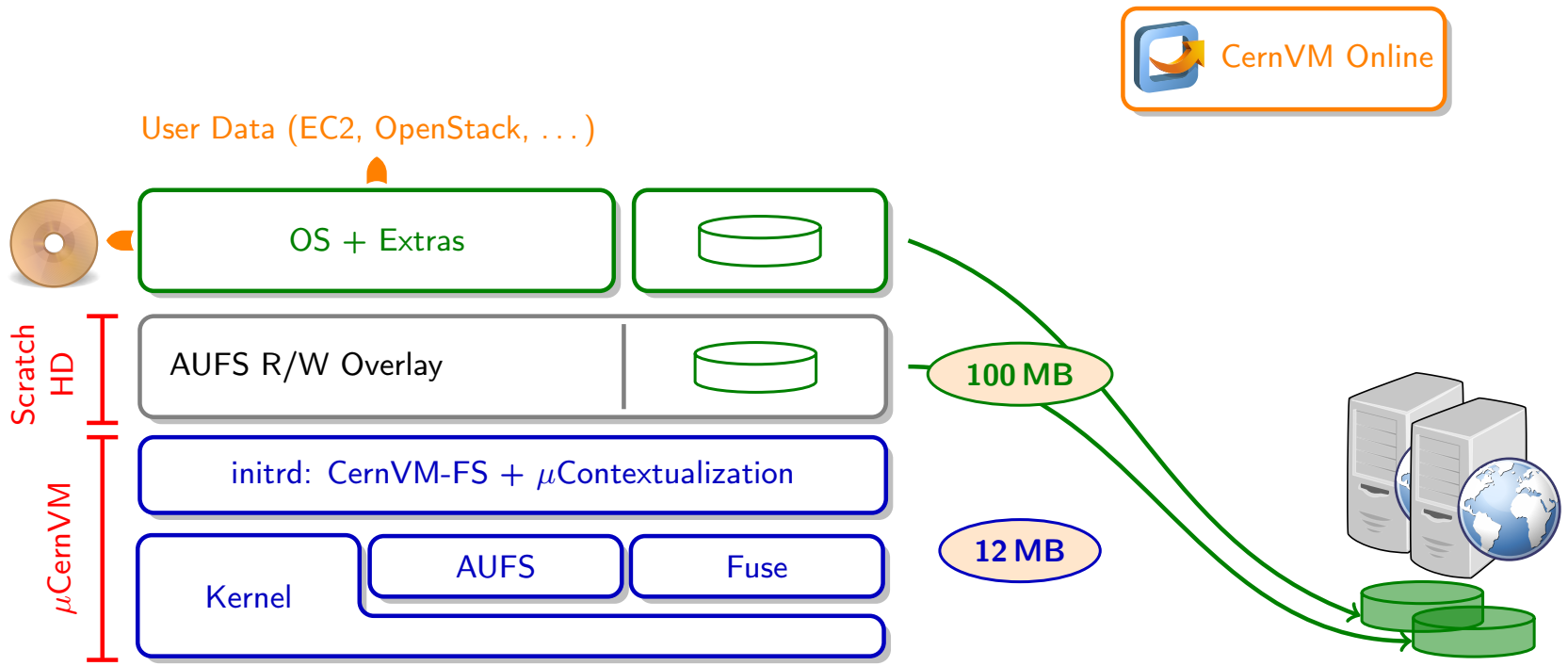
- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed
- Plain files

The *root hash* (40 characters) defines a file system snapshot (similar to git)

Track record of 5 years LHC software

Building blocks of CernVM

CernVM 3



Twofold system: μCernVM boot loader + OS delivered by CernVM-FS

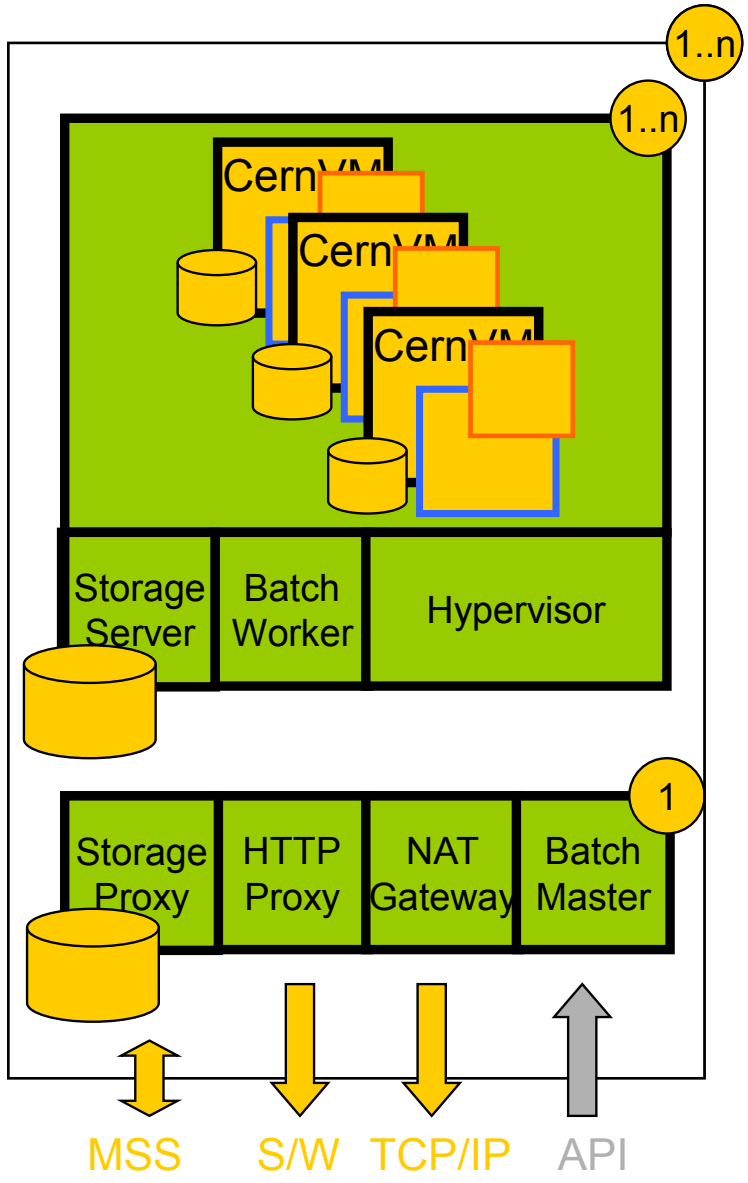
- The very same image can be *contextualized* to run Scientific Linux 4 32bit as well as the latest Scientific Linux 6 64bit
- $\approx$ 10 years with a single image

"Context"

- Small ASCII text snippets
- Can be versioned
- Human readable

Contextualization Examples

- Inject credentials (ssh, X.509)
- Condor head & batch services
- Squid server
- XrootD storage proxy
- Monitoring & directory service agents



Components of the CernVM Blue Print

Virtual Machine

- Linux distribution based on Scientific Linux.
- Supports all popular hypervisors.
- Minimal footprint, the VM *interface* is needed
- Flexible contextualization.

CernVM Filesystem

- Read-only, globally distributed file system optimized for software distribution.
- Based on plain files and HTTP
- Snapshotting and versioning file system
  - Already used in production by LHC experiments.

CernVM - based
data analysis environment
preservation

Bookkeeping

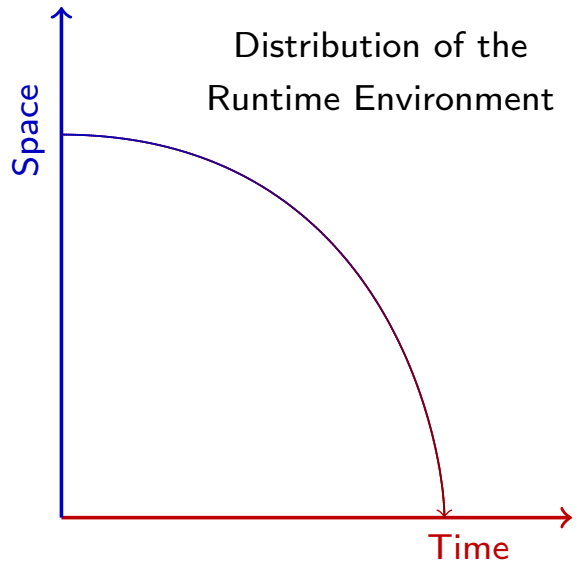
- CernVM-FS environment is defined by version strings. OS packages are defined by a versioned, closed package group (Meta-RPM)
- You need only the CernVM version string to rebuild CernVM image on demand.

Private Cloud

- Ensembles of CernVMs can recreate a virtual cluster for data processing.
- CernVM can be contextualized using a small subset of EC2 API that allows it to be deployed on public or private clouds

Summary

- Virtualization technology can easily **bridge tens of years**
- CernVM technologies provide a **data processing environment identified by a version string**
- Such virtual machines integrate well with today's cloud infrastructures
- Such virtual machines are easy to use and they can be given to "interested citizens"



Next Steps:

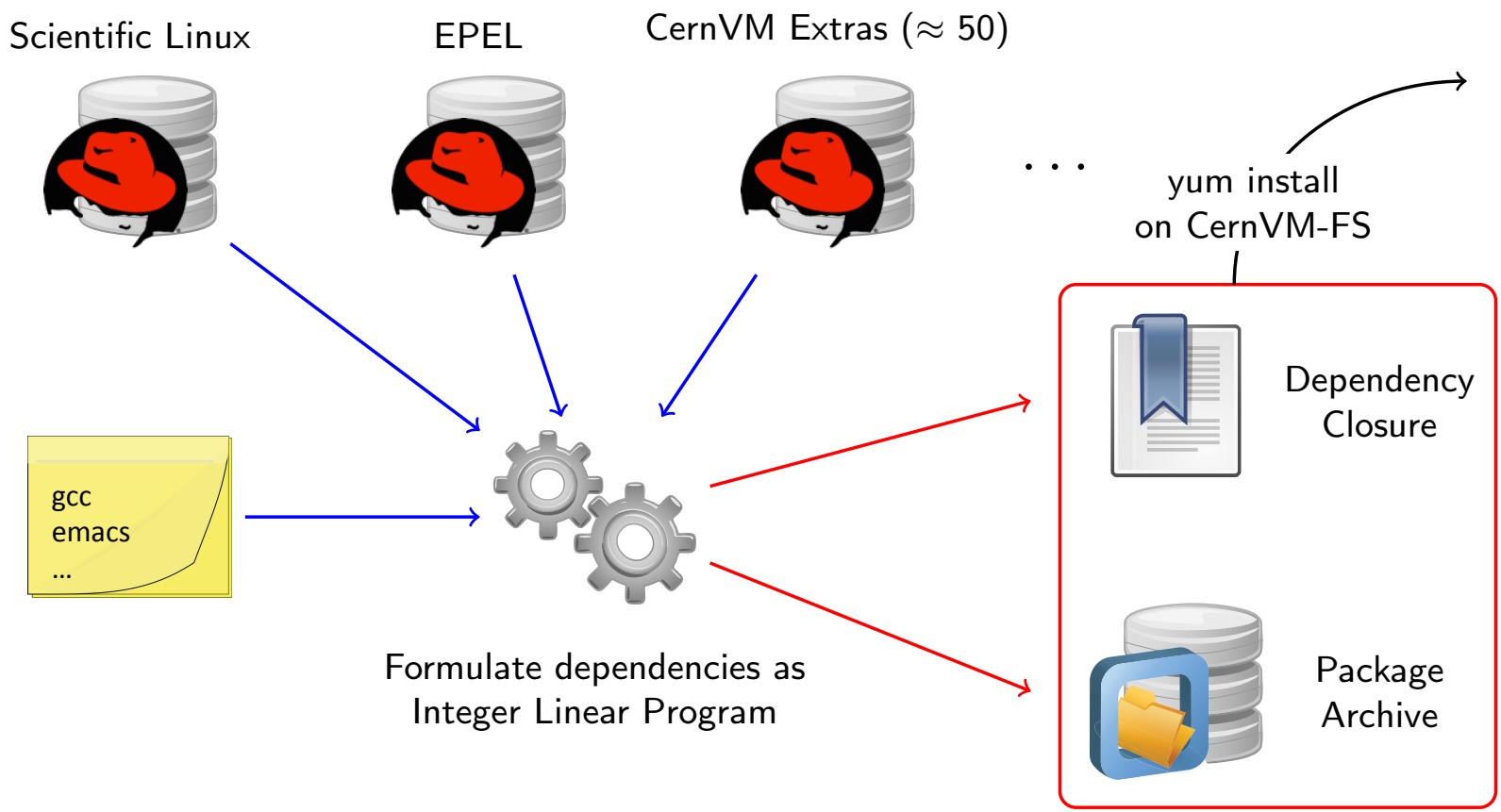
1. Investigate in virtualization and long-term software preservation outside HEP
2. Demonstrator: ALEPH physics on CernVM/Openstack

1. Backup Slides

Build Process: Scientific Linux on CernVM-FS

Maintenance of the repository **should not** become a Linux distributor's job
But: should be reproducible and well-documented

Idea: automatically generate a **fully versioned**, **closed** package list
from a "shopping list" of unversioned packages



Build Process: Package Dependency ILP

Normalized (Integer) Linear Program:

$$\text{Minimize } (c_1 \cdots c_n) \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{subject to } \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \leq \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Here: every available (package, version) is mapped to a $x_i \in \{0, 1\}$.
Cost vector: newer versions are cheaper than older versions.
(Obviously: less packages cheaper than more packages.)
Dependencies:
Package x_a requires x_b or x_c: $x_b + x_c - x_a \geq 0$.
Packages x_a and x_b conflict: $x_a + x_b \leq 1$.
(...)

Figures

≈17 000 available packages ($n = 17000$), 500 packages on "shopping list"
≈160 000 inequalities ($m = 160000$), solving time <10 s (`glpk`)
Meta RPM: ≈1 000 fully versioned packages, dependency closure

Idea: Mancinelli, Boender, di Cosmo, Vouillon, Durak (2006)

Hypervisor Support Status

Hypervisor / Cloud Controller	Status
VirtualBox	✓
VMware	✓
KVM	✓
Xen	✓
Microsoft HyperV	✓
Parallels	↯[4]
Openstack	✓
OpenNebula	✓[3]
Amazon EC2	✓[1]
Google Compute Engine	↯[2]

[1] Only tested with ephemeral storage, not with EBS backed instances
[2] Waiting for custom kernel support
[3] Only amiconfig contextualization
[4] Unclear license of the guest additions