Software Preservation

Jakob Blomer, CERN EP-SFT

3^{rd} DPHEP Collaboration Workshop, 21 June 2021

HEP Experiment Software Stacks and Environments

Analysis | Simulation | Reconstruction

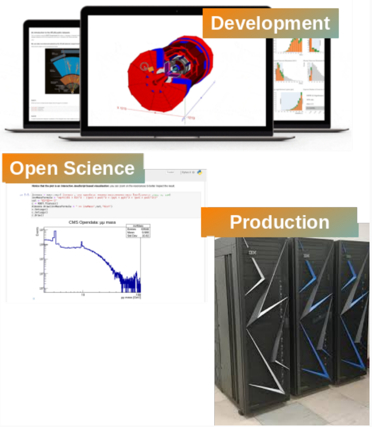
with ntuples

Experiment Software Framework:
EDM, Geometry, Conditions, . . .

HEP Libraries:
ROOT, Geant4, DD4hep, Generators, . . .

Data Access: XRootD, Davix, Arrow . . .

External Libraries: BLAS, Tensorflow, . . .
C++ Compiler, Python



HEP Experiment Software Stacks and Environments

Analysis | Simulation | Reconstruction

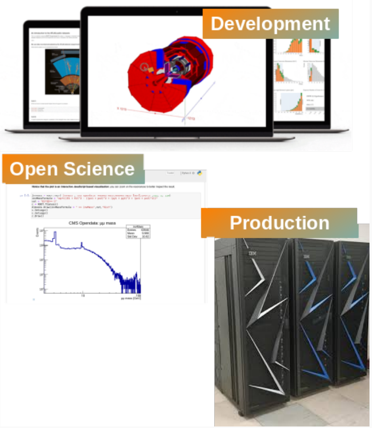
with ntuples

Experiment Software Framework:
EDM, Geometry, Conditions, ...

HEP Libraries:
ROOT, Geant4, DD

Example from LCG software stack combinatorics:
500 packages × 5 operating aystems ×
5 compilers × Python{2,3} × {opt, dbg}

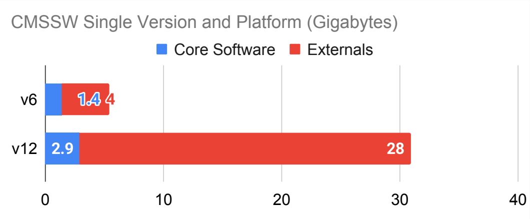
External Libraries: BLAS, Tensorflow, ...
C++ Compiler, Python



Growth of the Software

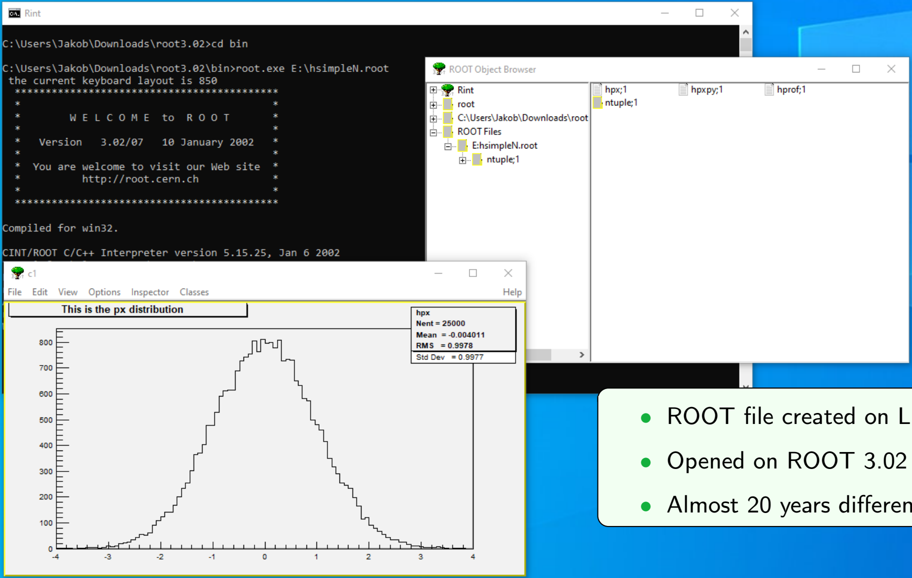
Compared to run 1-2, we now find

- Multiple target architectures: x86_64 micro-architectures (e. g. AVX512), AArch64, Power, GPUs
- A growing Python software ecosystem, in particular for machine learning tasks
- More agile software development: automated integration builds, nightly builds
- Generally we tend to add code and externals more often than removing software



My estimate: the software management problem for HL-LHC grows by a factor of 3-5.

Forward Compatibility in ROOT



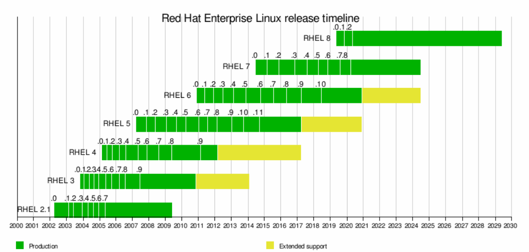
- ROOT file created on Linux with ROOT 6.24
- Opened on ROOT 3.02 on Windows 10
- Almost 20 years difference

Platform Lifecycle

- Our standard Linux platform, Red Hat Enterprise Linux (aka Scientific Linux, CentOS), has a life time of $\sim$10 years per release
- No security updates once out of maintenance, hence availability on central services stops (lxplus, lxbatch, . . .)

Two options for experiment application software to manage the operating system change

1. Porting & validation
   can be challenging wrt. legacy dependencies such as CERNLIB
2. Freezing & sandboxing
   Captures legacy software plus OS and compilers using virtualization technology



CernVM Technology

CernVM-FS
CernVM File System

Software and OS distribution to globally distributed HEP infrastructures

CernVM
Software Appliance

Portable VM/container for running and building LHC applications

- Software development team in the CERN EP-SFT group, operations teams at large Tier 1 sites
- Two products of the team: CernVM File System and CernVM Virtual Appliance

Example legacy software stacks on CernVM-FS: ALEPH, OPAL, DELPHI, NA48, CMS run 1

Benefits of the CernVM Technology

It is of course possible to freeze and sandbox with standard container/VM technology.
We think, however, that there are certain benefits in following the CernVM approach.

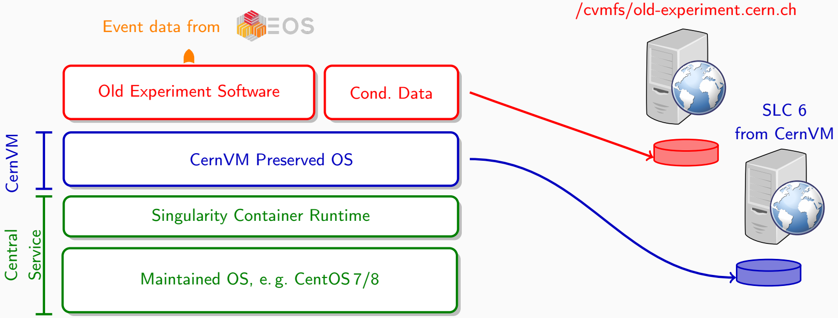
CernVM File System

- Production system for distribution of experiment software and conditions data, i. e. available on all CERN central services, the grid, most institutes
- Workflow for publishing identical between production and preserved software stacks
- Versioning is built-in, preservation part of regular software releasing
- Well-maintained location for sources, development environment, and binaries

CernVM OS Container

- Curated Linux platform with all dependencies to run LHC applications
- Frozen environments available for RHEL 4–6 platforms
- Aims at providing a minimal compatibility container (glibc + ϵ)

Sandboxing with CernVM and CernVM-FS I/II




```
[jblomer@lxplus7] singularity exec -B /cvmfs -B /eos \
>    /cvmfs/cernvm-prod.cern.ch/slc6 /cvmfs/old-experiment.cern.ch/application
```

Sandboxing with CernVM and CernVM-FS II/II

```
[jblomer@lxplus7] singularity exec -B /cvmfs -B /eos \
>    /cvmfs/cernvm-prod.cern.ch/slc6 /cvmfs/old-experiment.cern.ch/application
```

What it means

On the central lxplus7 service

1. start a Linux container (sandbox) using singularity
2. map /cvmfs and /eos from lxplus7 into the container
3. as a container operating system, use SLC6 provided by the CernVM repository
4. within this SLC6 sandbox, run preserved 'application' binary

Note: the sandbox should be seen as disconnected, all input data must come from a file system (CernVM-FS or EOS).
This prevents most security issues connected with running outdated software.

Containers and Data Preservation

- Decouples application from node operating system
- Allows for capturing a reproducible software environment as well as complete workflows
- Provides similar longevity than a static executable: ~ 20 years

- Facilitates a “black box” approach that impedes a proper understanding of the software stack inside
- Can be difficult to recuperate software sources and build environment

A possible approach to containers is using them as a minimal compatibility layer (glibc + ε), while the application universe comes from /cvmfs. We aim for this model with CernVM 5.

CernVM-FS as a Container Hub

/cvmfs/unpacked.cern.ch

- $>$ 1000 images
- $>$ 6.5 TB
- $>$ 95 M files

/cvmfs/singularity.opensciencegrid.org

- $>$ 630 images
- $>$ 2.5 TB
- $>$ 60 M files

Images are readily available to run with singularity,
including **base operating systems**, **experiment software stacks**, **explorative tools (ML etc.)**,
user analyses, and special-purpose containers such as **folding@home**

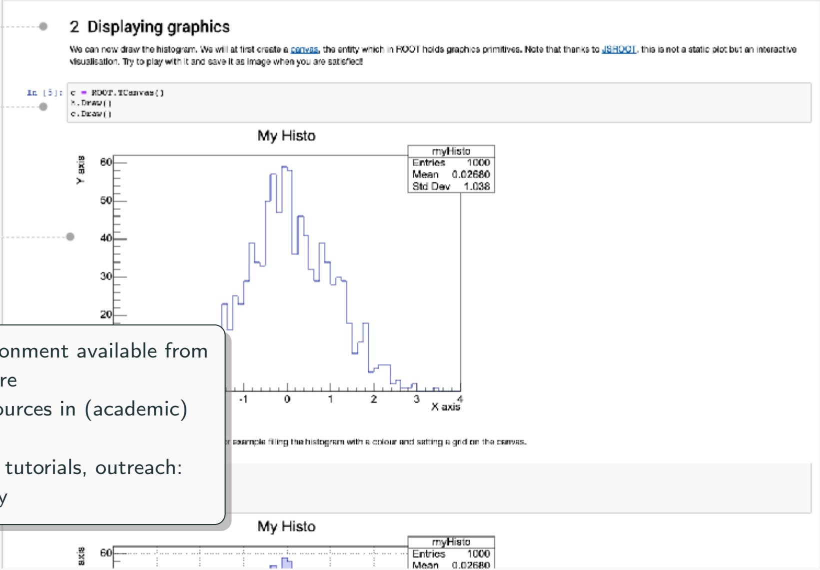
```
[jblomer@lxplus.cern.ch]$ singularity exec \
  '/cvmfs/unpacked.cern.ch/registry.hub.docker.com/library/debian:stable' \
  cat /etc/issue
Debian GNU/Linux 10 \n \l
```

Outreach: Jupyter Notebook Service for HEP

Text

Code

Graphics



- Data analysis environment available from all devices, anywhere
- Computational resources in (academic) clouds
- Great for teaching, tutorials, outreach: Tells a physics story

SWAN: Jupyter Notebook Service for HEP

UI/Core

Analysis platforms

Compute

- SWAN presents the familiar environment as a notebook
- A selection of software stacks readily available
- Data access via EOS home folder
- Can scale out computation to Spark
- Can be deployed on-premise with Science Box

Software

Storage

Infrastructure

Summary

- Continued investment needed in **software tooling**: build, packaging, test, and distribution tools: we need to manage and preserve source code + build environment + binaries
- Constructing software stacks becomes an **engineering discipline** ("software librarian")
  - Tendency to build complete stack including OS layer
    → completely independent from target node OS
  - Dedicated projects for turnkey stacks
    ▸ LCG ▸ Key4HEP
- **Forward and backward compatibility** needs to be a central part of our software and data formats
- We should define **reference architectures** and builds that are not subject to the complexity of extreme optimizations and rare hardware options

> Successful software preservation depends on proper tools and a certain discipline in maintaining software stacks; otherwise we risk to "capture the mess".

Long-Term Preservation Links

- CernVM software preservation in action: CMS run 1 software for SLC6
  http://opendata.cern.ch/record/252
- Paper: **“CERN Services for Long Term Data Preservation”** (2016) Download