Automatized Compilation of Large Packages Example of mpdroot

Ján Buša Jr.

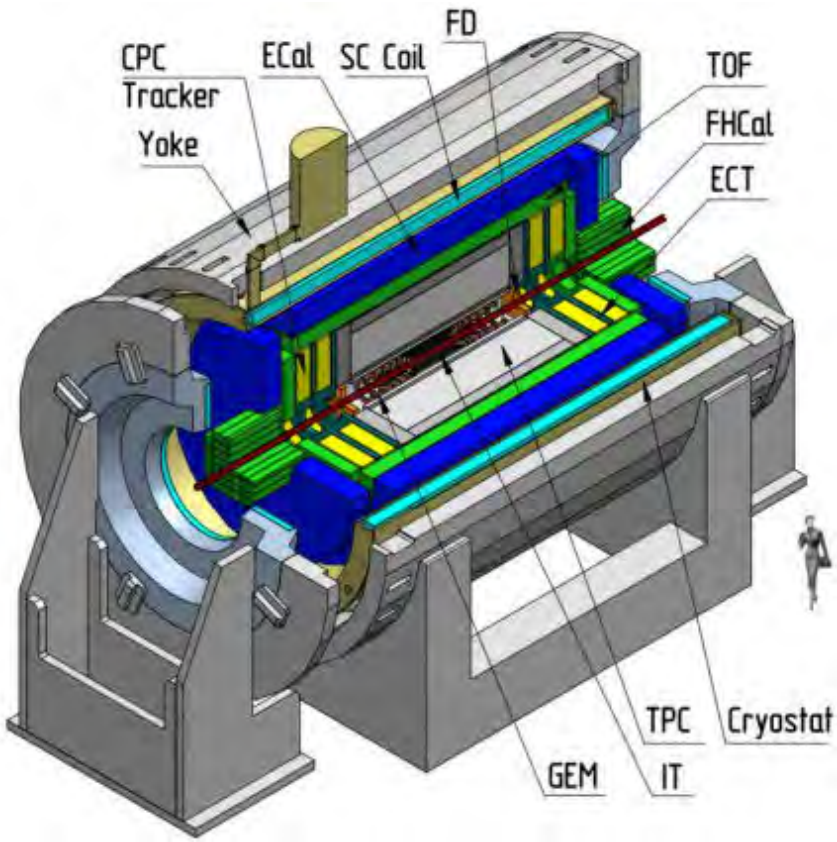
Mescheryakov Laboratory of Information Technologies, JINR, Dubna, Russia

Institute of Experimental Physics, SAS, Košice, Slovakia

Dubna
15. 11. 2022

Multi-Purpose Detector

- part of **N**uclotron-based **I**on **C**ollider f**A**cility
- **M**ulti **P**urpose **D**etector – main detector under costruction
- mpdroot – library for reconstruction of tracks and physical analysis (`http://mpdroot.jinr.ru/`)



Just Install FairRoot and Build mpdroot...

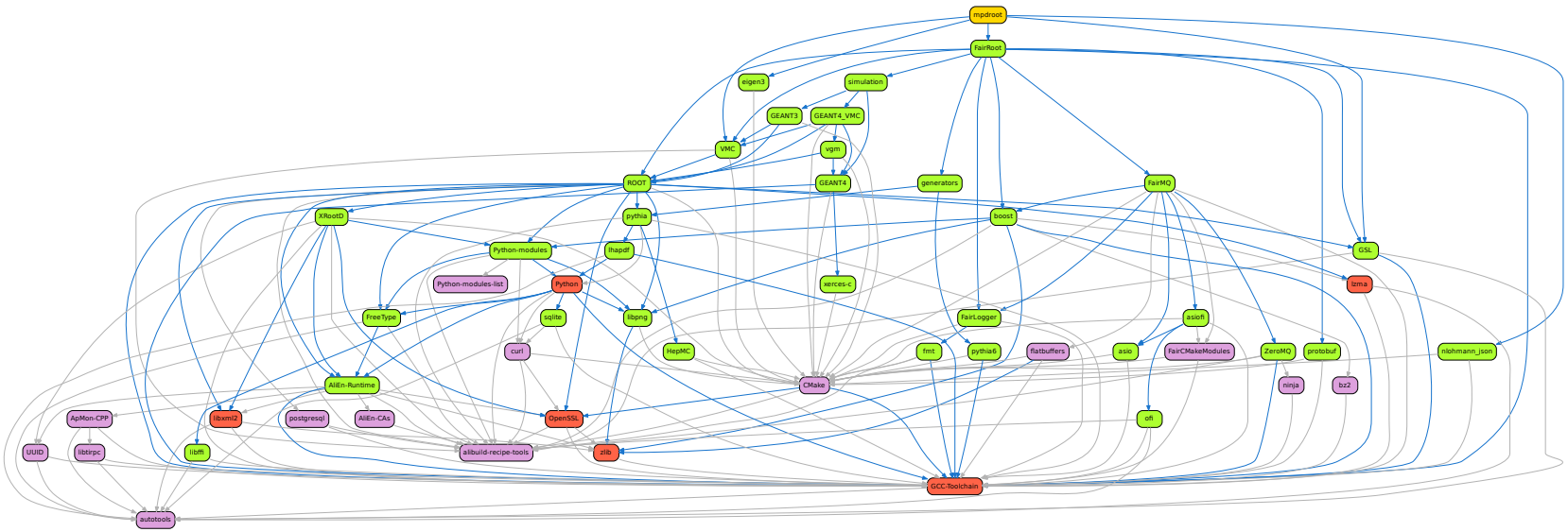
How to install MPDroot (each user had to build own version):

- install all basic packages (about 70);
- download, patch, and build FairRoot;
- download and build mpdroot from sources. In-source build highly recomended.

It's really "easy":

- basic packages are often not available in compatible versions
- patching FairRoot can be tedious (during last major version change more than 100 lines of code)
- building FairRoot takes about 2−4 hours on a regular computer
- in-source build guides users to use improper paths when linking own code
- users tend to use same version of mpdroot for long time

The mpdroot Dependencies



Centralized Build of MPDroot

Requirements:

- we want to deliver latest version of MPDroot to the users
- all dependencies have to be statisfied and no version clashes are allowed
- results need to be repeatable

Solution:

- CVMFS – robust distribution of software not only between clusters/supercomuters
- aliBuild – set of tools for building software together with its dependencies
- toolbox – rootless wrapper for podman (docker). Supported on most linux-based systems. On MacOS and Windows one can use docker
- EnvironmentModules – environment enabling loading multiple versions of the same software with its dependencies

CernVM-FS
https://cernvm.cern.ch/fs/

- CMVMFS is a **read-only** file system
- anyone can mount the shared directories to its computer
- extensively uses caching of the files on local computer
- only used (accessed) files are downloaded. That makes first load slow but following are as fast as local usage
- since it is centralized, changes on server are directly available to all users

Toolbox
https://github.com/containers/toolbox

- wrapper for podman (rootless docker alternative) – ordinary user can use it
- allows user to "work" inside virtual machine – no connection to the users system
- we support CentOS7, but to test our software we build also CentOS Stream 8 and 9 versions
- basic system image is fixed without any updates and therefore all results are reproducible

Environment Modules
https://modules.readthedocs.io/

- started as a project to separate various versions of same software
- simple "module files" contain paths to the software together with list of dependencies
- to use some software one has to load it first via command `module add mpdroot/latest`. Paths to executables as well as libraries is than added in the commandline. All necessary dependencies are loaded as well.

Example of module file (excerpt):

```
module load BASE/1.0 GEANT4/$GEANT4_VERSION-$GEANT4_REVISION
module load ROOT/$ROOT_VERSION-$ROOT_REVISION
setenv MPDROOT \$PKG_ROOT
prepend-path ROOT_INCLUDE_PATH \$PKG_ROOT/include
prepend-path LD_LIBRARY_PATH \$PKG_ROOT/include
```

aliBuild
https://alisw.github.io/alibuild/

- created as a solution to the ALICE experiment software distribution problem
- automatic build of software and all its dependencies
- reusing already built packages
- if some package was changed, all dependants are rebuild
- apropriate modules are created and numbered automatically
- can get messy if packages change frequently and are not cleared
- based on recipes – guides how to build some package

mpdroot Recipe

```
package: mpdroot
version: "%(commit_hash)s"
tag: "dev"
source: https://git.jinr.ru/nica/mpdroot.git
requires:
  - FairRoot
  - eigen3
  - nlohmann_json
  - VMC
  - GSL
---
unset SIMPATH
export MPDROOT=$INSTALLROOT

cmake $SOURCEDIR                                                              \
      ${BOOST_ROOT:+-DBOOST_ROOT=$BOOST_ROOT}                                 \
      ${EIGEN3_ROOT:+-DEIGEN3_ROOT=$EIGEN3_ROOT}                              \
      ...
      ${LIBXML2_ROOT:+-DPC_LIBXML_LIBDIR=$LIBXML2_ROOT/lib}                   \
      ${NLOHMANN_JSON_ROOT:+-DNLOHMANN_JSON_ROOT=$NLOHMANN_JSON_ROOT}         \
      -DCMAKE_INSTALL_PREFIX=$MPDROOT
cmake --build .  -j$JOBS install

# Modulefile
MODULEDIR="$INSTALLROOT/etc/modulefiles"
MODULEFILE="$MODULEDIR/$PKGNAME"
mkdir -p "$MODULEDIR"
alibuild-generate-module --bin --lib > $MODULEFILE
cat >> "$MODULEFILE" <<EoF
setenv MPDROOT \$PKG_ROOT
setenv MPDROOT_MACROS \$PKG_ROOT/macros
prepend-path ROOT_INCLUDE_PATH $::env(BASEDIR)/GSL/$GSL_VERSION-$GSL_REVISION/include
EoF
```

Changing Defaults

```
default:
  - &MPDROOT_TAG v22.09.22
  - &GCC_TOOLCHAIN_TAG v10.2.0-alice2
package: defaults-release
version: v1
disable:
  - arrow
env:
  CMAKE_BUILD_TYPE: RELWITHDEBINFO
  CXXFLAGS: -fPIC -O2 --std=c++17
  ENABLE_VMC: 'ON'
overrides:
  mpdroot:
    tag: *MPDROOT_TAG
    version: *MPDROOT_TAG
  ROOT:
    tag: "v6-26-08"
    requires:
      - GSL
      - libxml2
  GCC-Toolchain:
    tag: *GCC_TOOLCHAIN_TAG
    version: *GCC_TOOLCHAIN_TAG
---
```

Wrapping Up

Our view:

- software is stored in git
- on each push to master or creation of tag, command `aliBuild --build --no-deps --defaults releaseName` is called and built package is placed into CVMFS. Software is available no later than 1h after being sucessfully built.

Developer's view:

- `toolbox enter c7-nica`
- `source /cvmfs/nica.jinr.ru/sw/login/login.sh`
- `module add mpddev/latest`
- `export MPDROOT=/path/to/install`
- `git clone path_to_mpdroot && mkdir build && cd build && cmake .. && make install`

Wrapping-Up (Continued)

User's view:

- `toolbox enter c7-nica`
- `source /cvmfs/nica.jinr.ru/sw/login/login.sh`
- `module add mpdroot/latest`

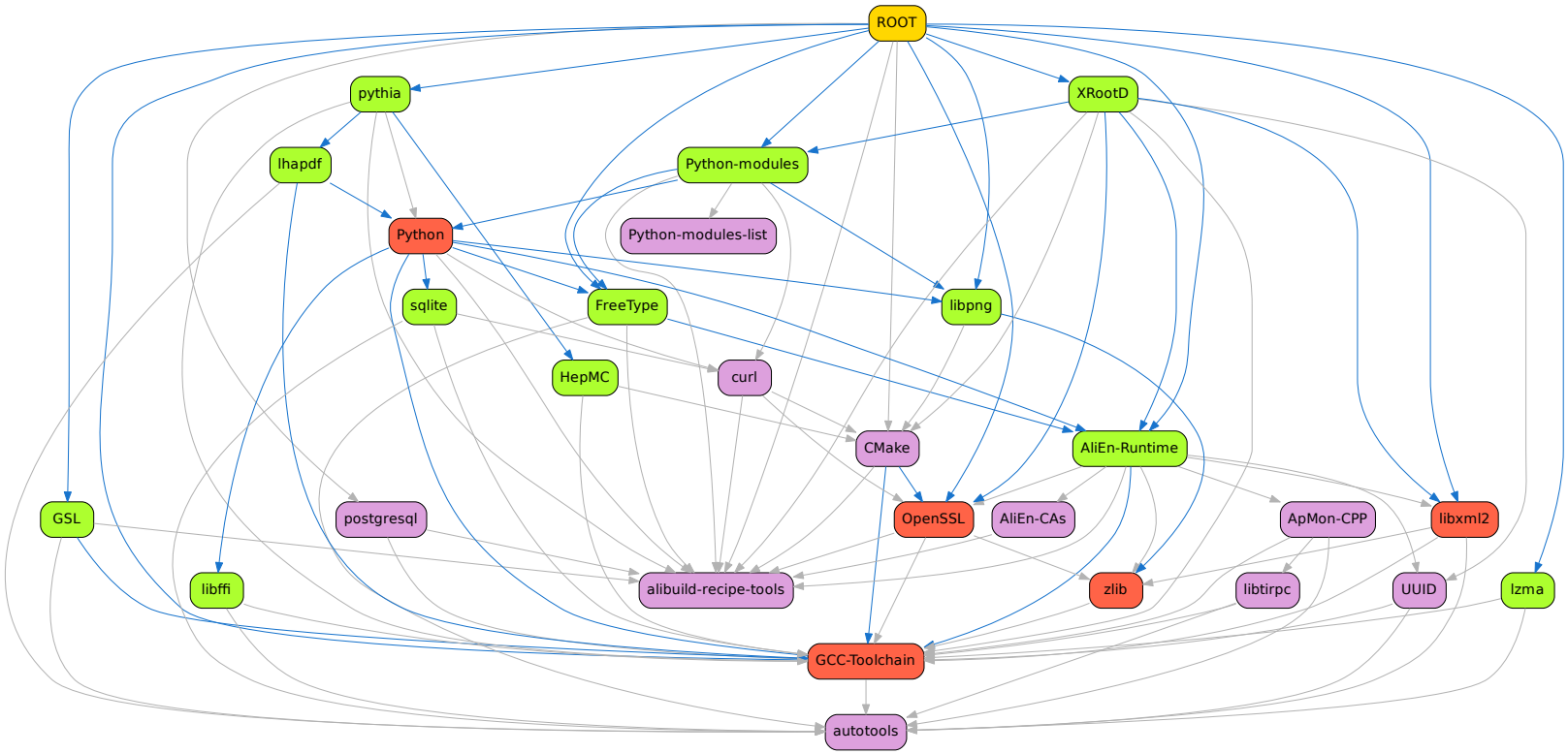
or concrete version

- `module add mpdroot/v22.09.22-1`

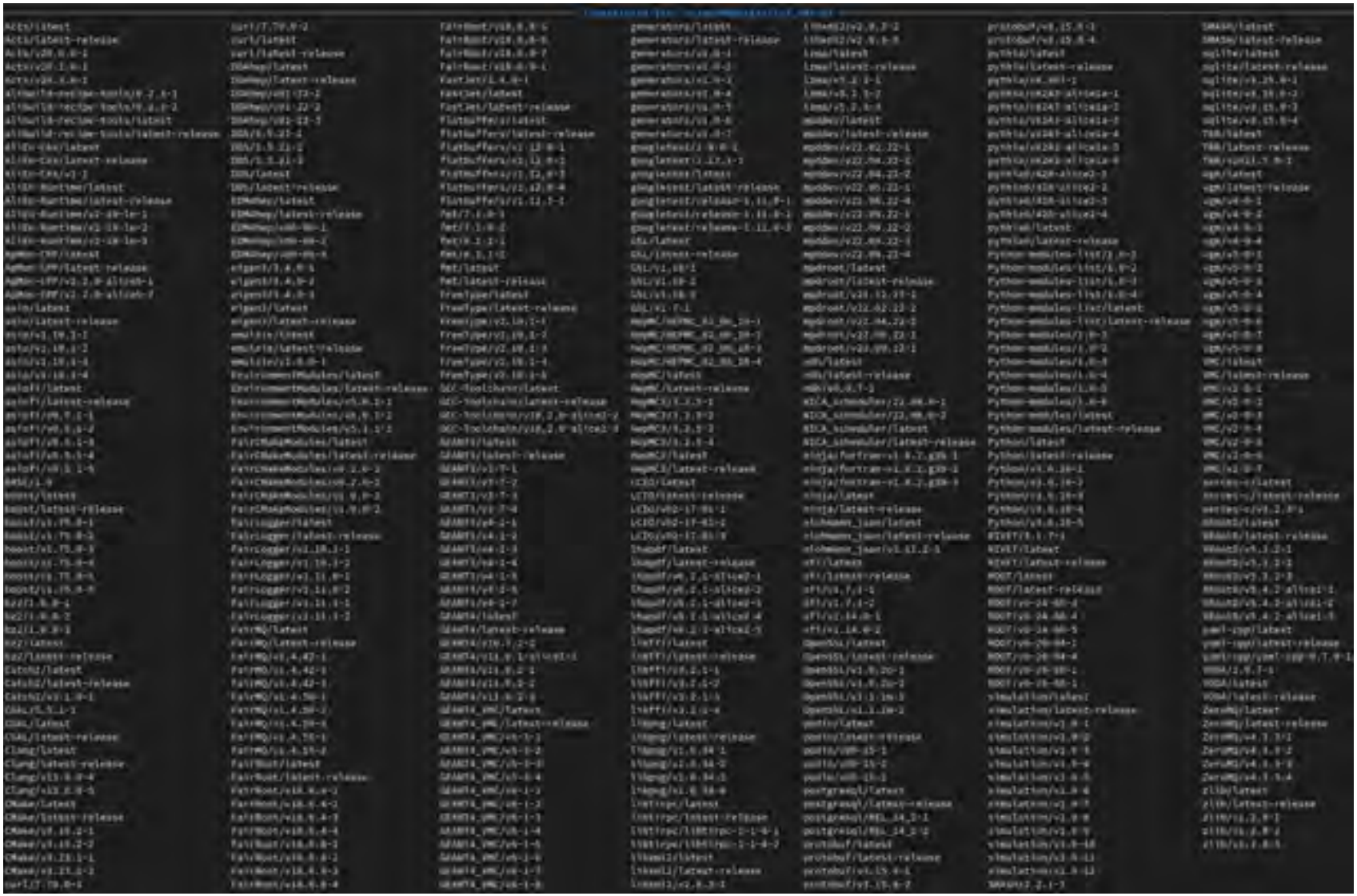
Conclusions

- for ordinary users build of mpdroot takes no time
- latest version of software is instantly available to everyone
- it is easy to return to previous versions (including dependencies)
- all results are repeatable since we store basic system image as well as all recipes
- most of the packages we build from scratch (including low level, s.a., CMake, GCC, OpenSSL)
- with recipes new dependencies are distributed among developers without necessity for them to install anything (it is done automatically)

All Great Except. . .



It can't be that bad, can it?



Help Needed

- (semi)automatic cleaning of obsolete packages needs to be added
- GUI for cleaning of packages would be nice to have
- accounting – which packages were used and what is the distribution of their usage in time
- distribution of mpdroot (and dependencies) via rpms
- protection of closed-source software
- sometimes even new recipes are not easy to add

Bachelor/Master/...

What I need to know:

- nothing – kind of

What will I get:

- diploma

What will I learn (part of/maybe all):

- git (not only for storing files)
- bash, CMake, aliBuild/aliDist
- Docker/Toolboxes, RHEL based packaging
- creating web-based application
- LaTeX