Everything you always wanted to know about CbmRoot

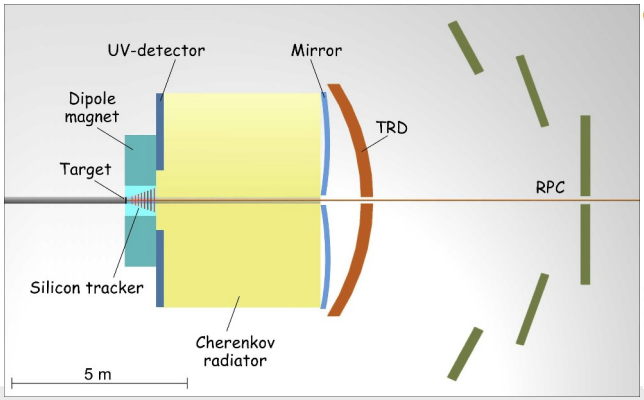
But were afraid to ask

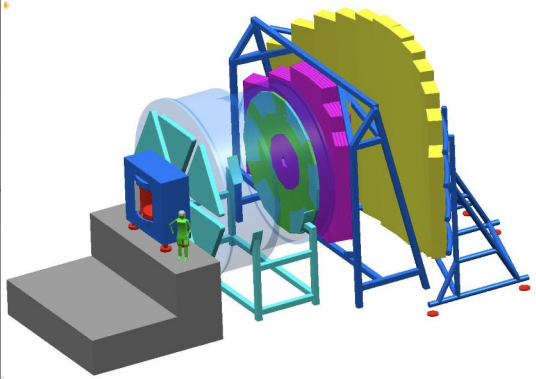
Outline

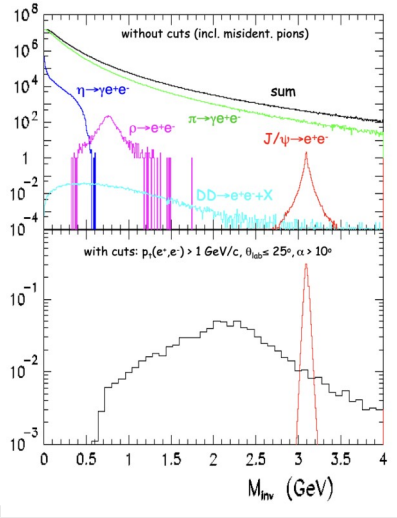
- History
  - Already more than 20 years of CbmRoot
- Infrastructure behind CbmRoot development
  - Tools
- Workflow
  - How to get changes into CbmRoot
- Summary and Outlook

The Beginning

- During the early development phase there was no separation into FairRoot and CbmRoot
  - 2002 two meetings (1, 2) at GSI took place to discuss the possibility of an experiment which should study dense baryonic matter using the planned new accelerator
  - The first CBM Collaboration Meeting was in January 2003
- CBM needs simulations to prove that the planned experimental setup works for the physics they are after
- First implementation in Geant4







The Pregnancy

- In the first CBM collaboration meeting (Jan. 29, 2003) the future FairRoot is discussed
  - Exploration of Virtual Monte Carlo (VMC)
    - D. Bertini and M. Al-Turany (GSI)
  - Also D. Kresan and R. Karabowicz were present
  - All 4 are now working for GSI IT
- In the second CBM collaboration meeting (Jul. 7-8, 2003) the first presentation about the VMC based framework is given
  - The simulation tools VMC and Geant4 (D. Bertini, GSI)
  - Unfortunately the presentation is lost

The Birth

- During the third meeting (Feb. 11-13, 2004) the new simulation framework was presented the first time
  - Status of the CBM simulation framework (M. Al-Turany, GSI)
- The first official release was done only some days later at 10.03.2004 in a CVS repository
  - We can celebrate more than 20 years of CbmRoot and also FairRoot !!!!
- Development before was done in private repositories which are probably lost
- All developments since then are accessible from the different repositories
  - CVS repositories are not freely accessible

General requirements

- Software should be usable in short time due to deadlines for CBM (LOI, TDRs, …)
  - Only two developers
  - Impossible to write everything from scratch
- Easy to use
  - Users are physicists and not experienced developers
- Easy to maintain
  - Reuse existing tools and developments
  - Keep the code base as small as possible
- Easy to install
  - Don’t depend on a specific OS or compiler

Motivation

- CBM needs a reliable simulation environment in short time (deadlines for LOI, TDR, …)
- Limited manpower (two developers)
- Reuse as much as possible existing code and tools
- Profit from development done for other experiments
  - ALICE: VMC, Eve
  - Hades: ASCII geometry interface, parameter management
- Which MC transport simulation to use
  - One would like to use the modern and maintained GEANT4
    - lack of knowledge about GEANT4 (intrinsic cuts / physics list …)
    - At that time it was extremely difficult to get support for working with Geant4
  - Better knowledge of “old” MC’s: GEANT3, FLUKA

Design Decisions I

- Immediate code release
  - Get feedback and bug fixes very fast
- Modular code
  - Allow to change algorithms and detector description easily
- Keep the code base minimal
  - Only implement what is really needed and not what potentially could be used in 20 years
- Reuse existing standard tools if they do the job
  - Build system
  - Code documentation
  - Code management

Design Decisions II

- Performance and stability has highest priority
  - Avoid fights about idealistic views and design concepts
    - Use C-Arrays where appropriate
    - Use TClonesArray for data instead of STL containers
      - Speed
      - Serialization of data to file
- Reuse existing code and concepts
  - Basic ideas and structures from AliRoot
  - Geometry and parameter interfaces from Hades
  - Virtual Monte Carlo for simulation
  - ROOT geometry package for navigation
  - Extent ROOT TTask for reconstruction and analysis

Main features I

- Using VMC for the simulation allows
  - Running different transport MCs from the same application
  - No need to change any user code (detector response, IO, physics settings) for a specific transport model
  - Describe the geometry only once and use it for any transport MC
    - Convert it to native geometries of the transport engine if needed/wanted
    - Allows to use the navigation from ROOT or the transport engine
- Dynamic event structure
  - Detectors can be added/replaced/removed without changing the code
  - Simple analysis can be done in ROOT without loading additional libs
  - Transient (in memory) and persistent (on file) objects are the same
  - ROOTs TTree friend mechanism allows to connect information of different files without copying and data from file to file

Main features II

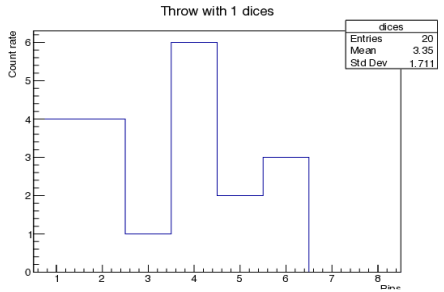
- Don’t create a binary
  - Use the ROOT binary and ROOTs plug-int mechanism to dynamically load libraries when needed
  - Meanwhile there are also some binaries, mainly for online
- Don’t use a fixed setup or run configuration
  - Use ROOT macros to define the experimental setup or the list of tasks for a reconstruction or analysis
  - Also use ROOT macros read at runtime to set Geant3 or Geant4 configurations
- These two features enables the user to switch very flexible between experimental setups and configurations
  - Change macro and rerun

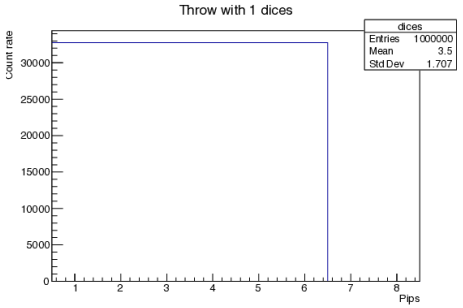
Interlude: Monte Carlo Simulation

- Monte Carlo (MC) simulations are a mathematical method to approximate the possible results of an uncertain event
- MC simulations are used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables
- A MC simulation consist of the following three steps
  - Modelling: Create a model which describes the random variables and their connection
  - Simulation: Do many iterations of the simulation by exchanging the random variables by random numbers
  - Analysis: Analyse the results of the simulation to extract the probability of different results

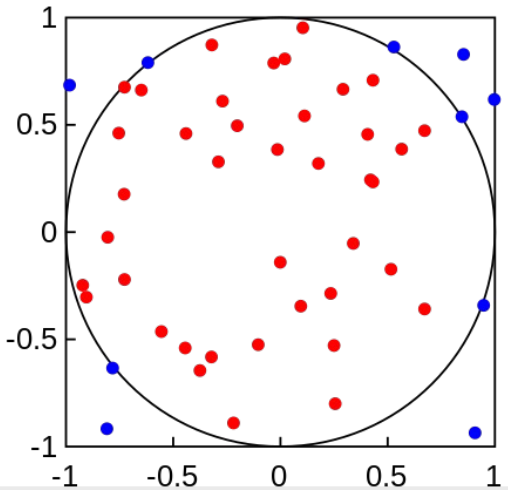
Interlude:
Monte Carlo Simulation Examples

- Throwing a dice
  - Result can be any of the numbers 1-6 → Random variable
  - Model the dice
    - Draw a random number between 1-6
      - Result can be any of the numbers 1-6
  - Redo the simulation many times and plot the results
    - Probability for any of the numbers will be ~1/6





- Calculate π
  - The size of the circle to the size of the box is like the number of random points inside the Circle to the total number of points

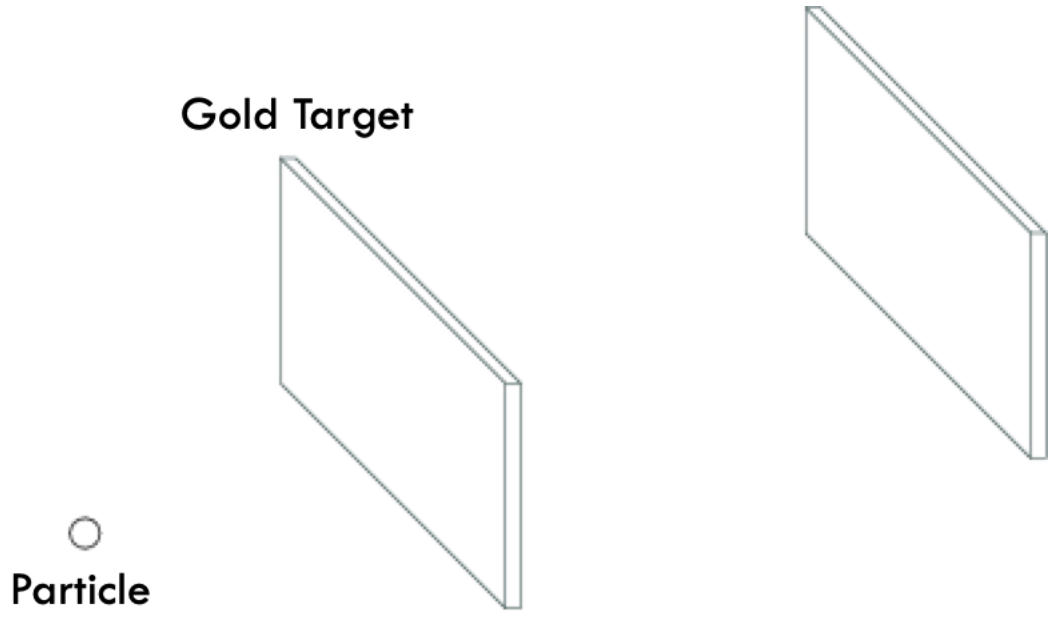


$$\frac{Area(Circle)}{Area(Box)} = \frac{\pi \cdot r^2}{(2 \cdot r)^2} = \frac{\pi}{4} \quad \Rightarrow \pi = \frac{4 \cdot Points(Circle)}{Points(Total)}$$

Interlude: Monte Carlo Transport Simulations

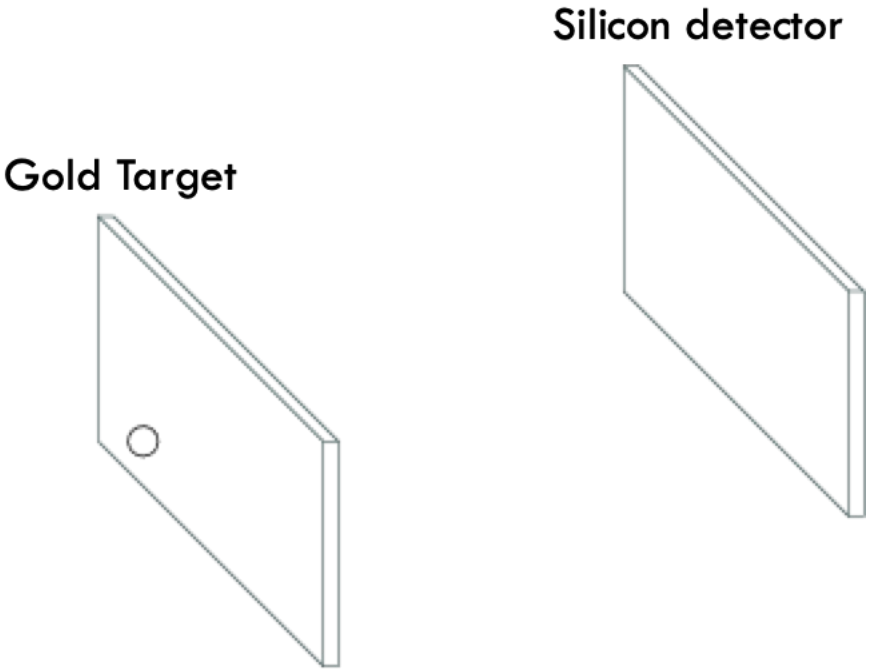
- Simulation software designed to describe the passage of elementary particles through matter, using MC methods
- Create a model of the experimental geometry and the physics processes
- Simulate the passage of many particles through the geometry
- Analyse the intersections of the particle trajectories with the active parts of the detector geometry

Interlude: MC Transport in Detail I



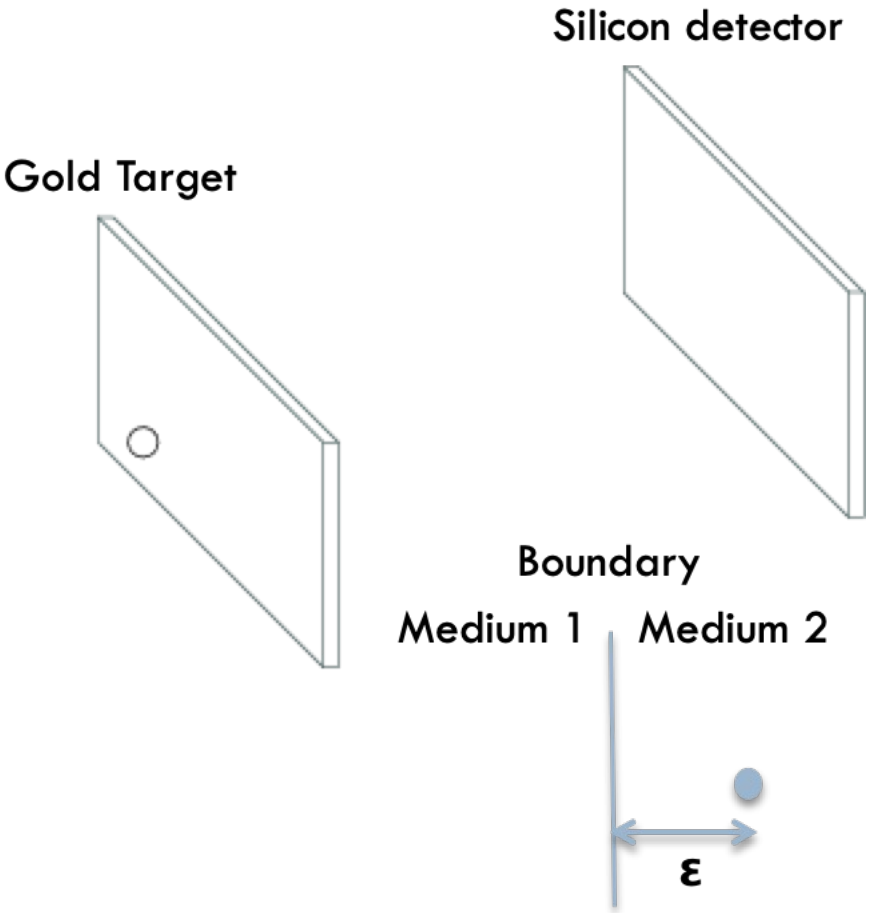
- Build model of geometry and physics
- Put one particle at position (x, y, z) with momentum (Px, Py, Pz)
- Calculate the distance to the next boundary in direction of particle track
- Calculate the distance where the next interaction in the current medium happens
- Depending on the medium, the particle and the physical settings there are several physical processes to be taken into account

Interlude: MC Transport in Detail II



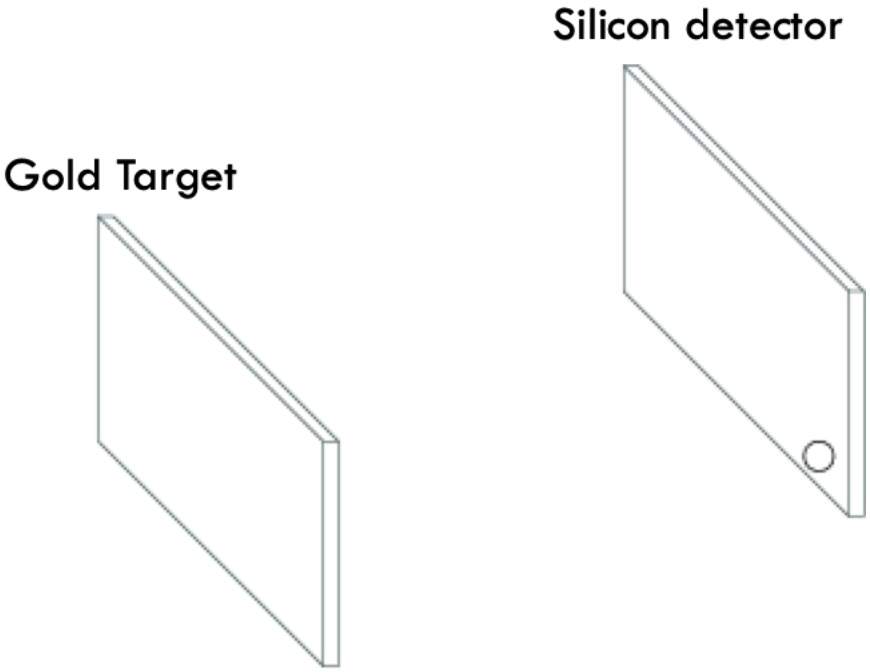
- **Particle has no physics interaction till the next boundary**
- **Transport simulation moves particle to the next boundary**

Interlude: MC Transport in Detail III



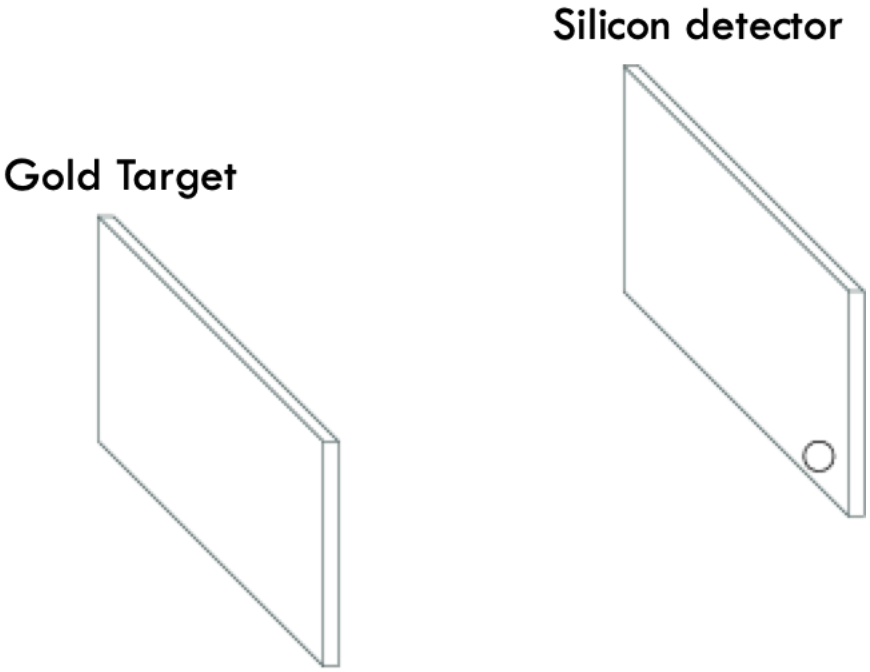
- Particle has no physics interaction till the next boundary
- Transport simulation moves particle to the next boundary (target)
- Due to problems with mathematical precision the particle is moved slightly inside the new volume
- In the target again the distance to the next boundary and to any physics interaction is checked
- If an interaction in the target happens the particle is moved to the position of the interaction and the particle properties are updated
  - Scattering changes the momentum and such the direction of the particle
- Repeat two previous steps till the particle leaves the volume

Interlude: MC Transport in Detail IV



- Particle has no physics interaction till the next boundary
- Transport simulation moves particle to the next boundary (silicon detector)
- Since the detector is an sensitive part of the detector some user code is called for every step in the sensitive volume
- Allows to access detailed information about the particle like
  - Position
  - Momentum
  - Energy
  - Energy Loss
  - Particle ID
  - ...

Interlude: MC Transport in Detail IV

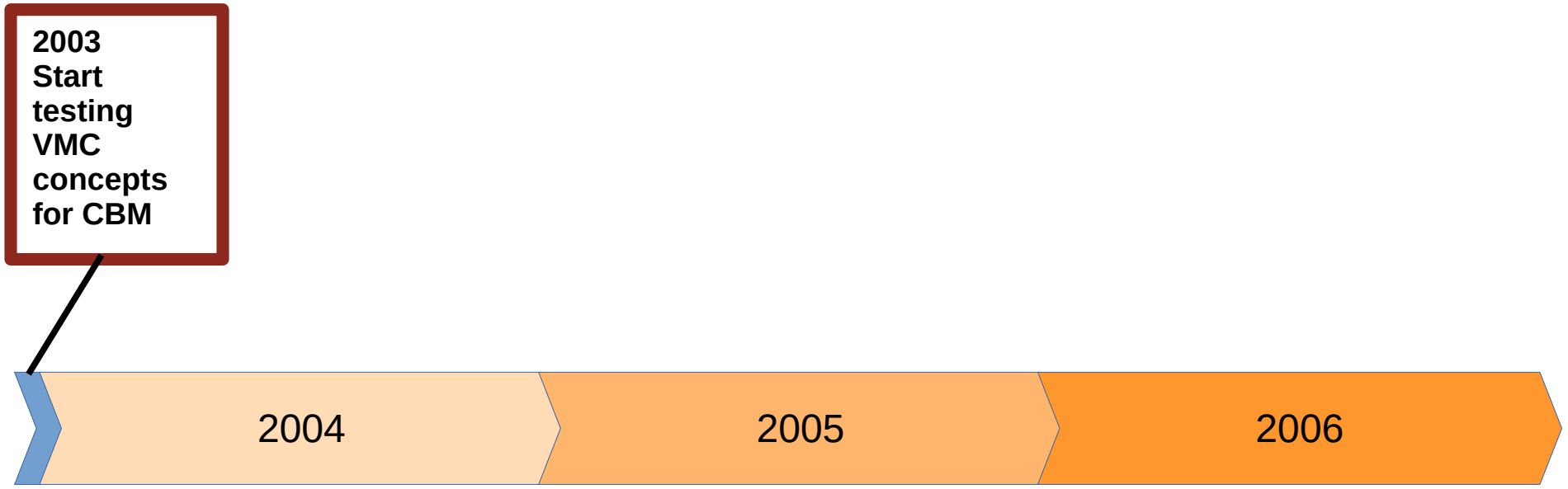


- Particle has no physics interaction till the next boundary
- Transport simulation moves particle to the next boundary (silicon detector)
- Since the detector is an sensitive part of the detector some user code is called for every step in the sensitive volume
- Allows to access detailed information about the particle like
  - Position
  - Momentum
  - Energy
  - Energy Loss
  - Particle ID
  - …
- Follow the particle until it is stopped, it disappears or it crosses the boundary of the experiment

Interlude: Virtual Monte Carlo

- Virtual Monte Carlo is a software framework that enables the simulation of different MC transport simulations without modifying the user code
- Decouples the user code from the concrete MC transport simulation
- Allows to use different MC transport simulations
- Currently the following three are supported by VMC
  - Geant3 (Fortran, 1982-2000)
  - Geant4 (C++, 1998-2024)
  - Fluka (Fortran, ?-2024)
- Geant: Geometry and Transport
- Fluka could not be used to licensing issues

Compressed History

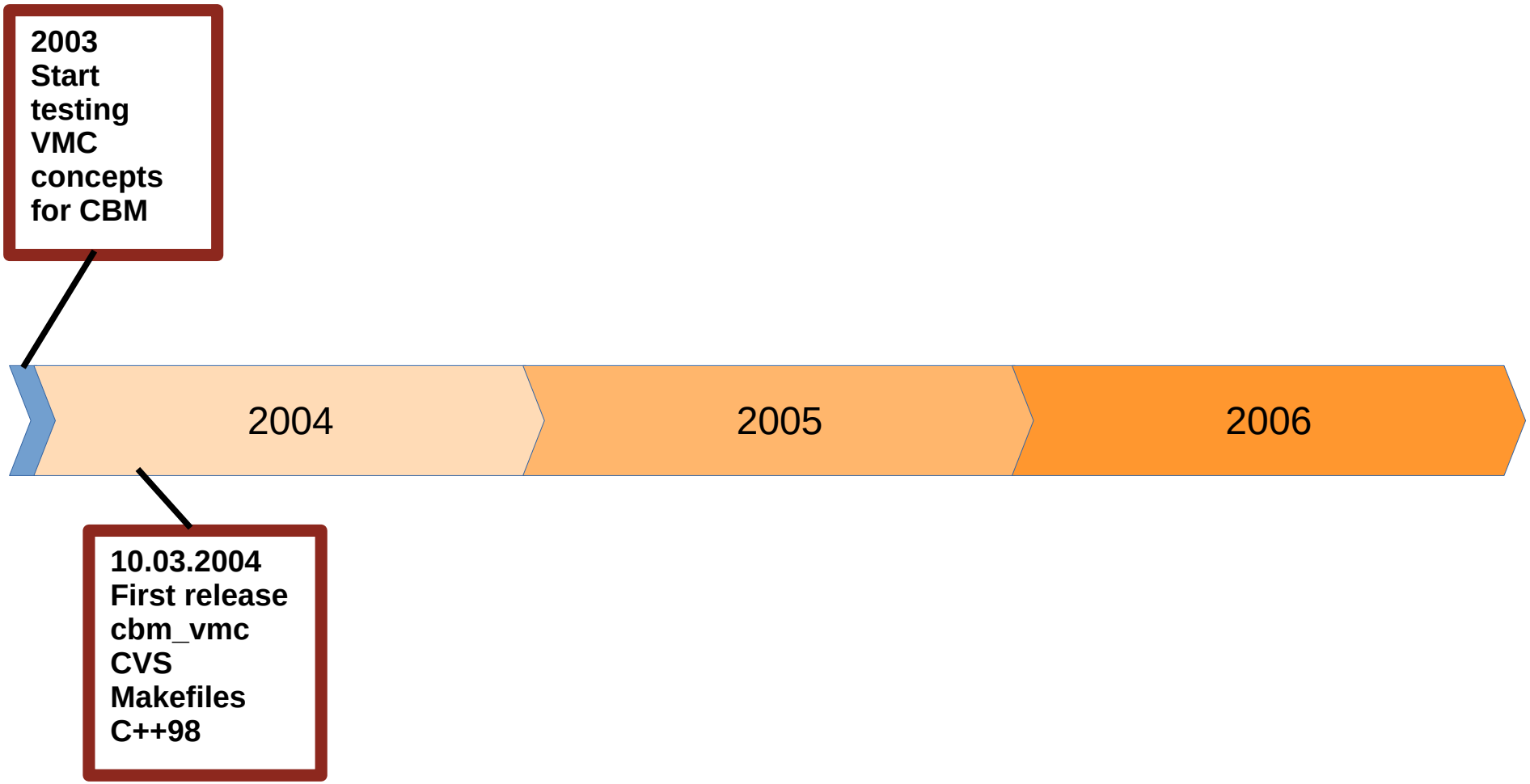


Compressed History

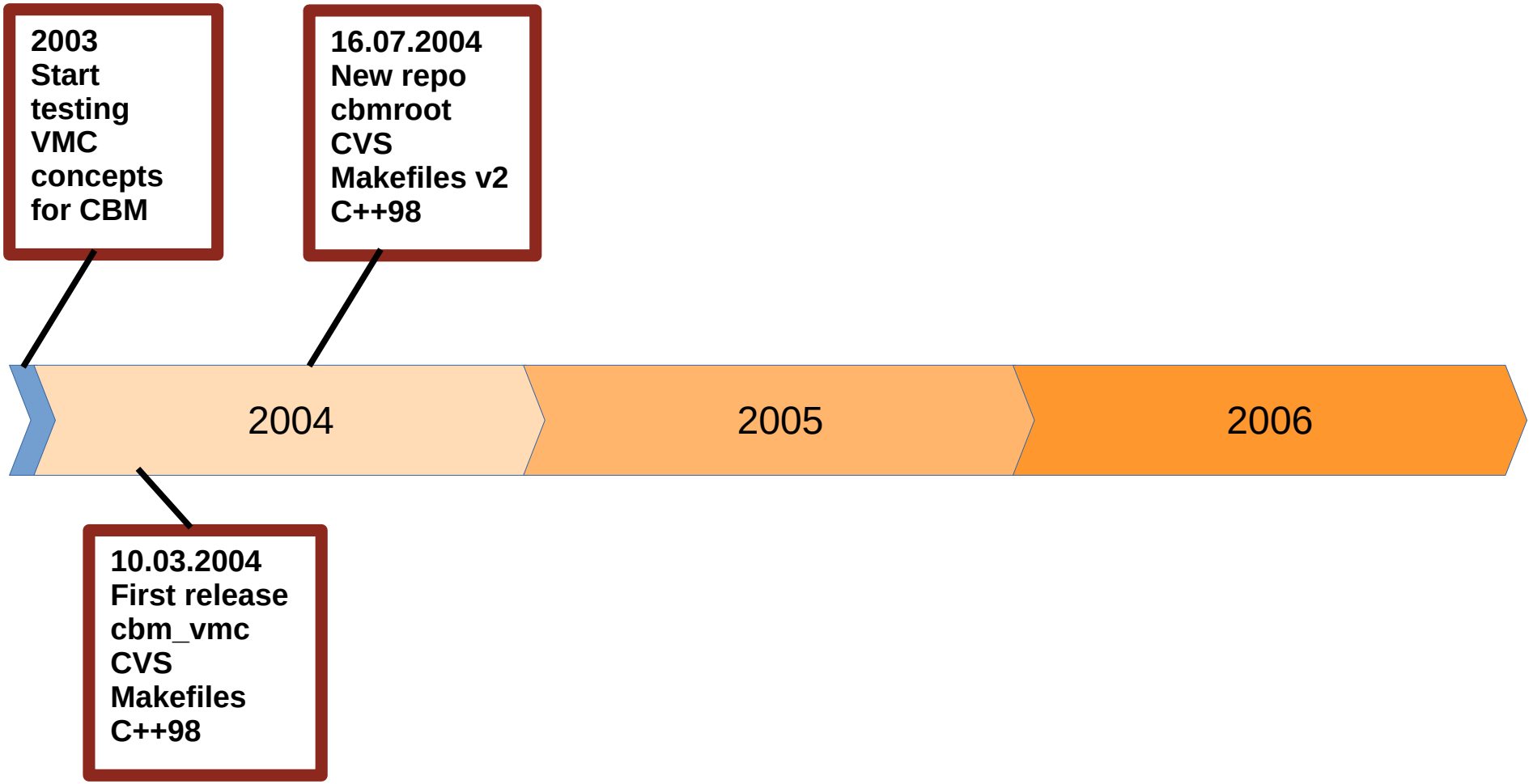
2003
Start testing VMC concepts for CBM

2

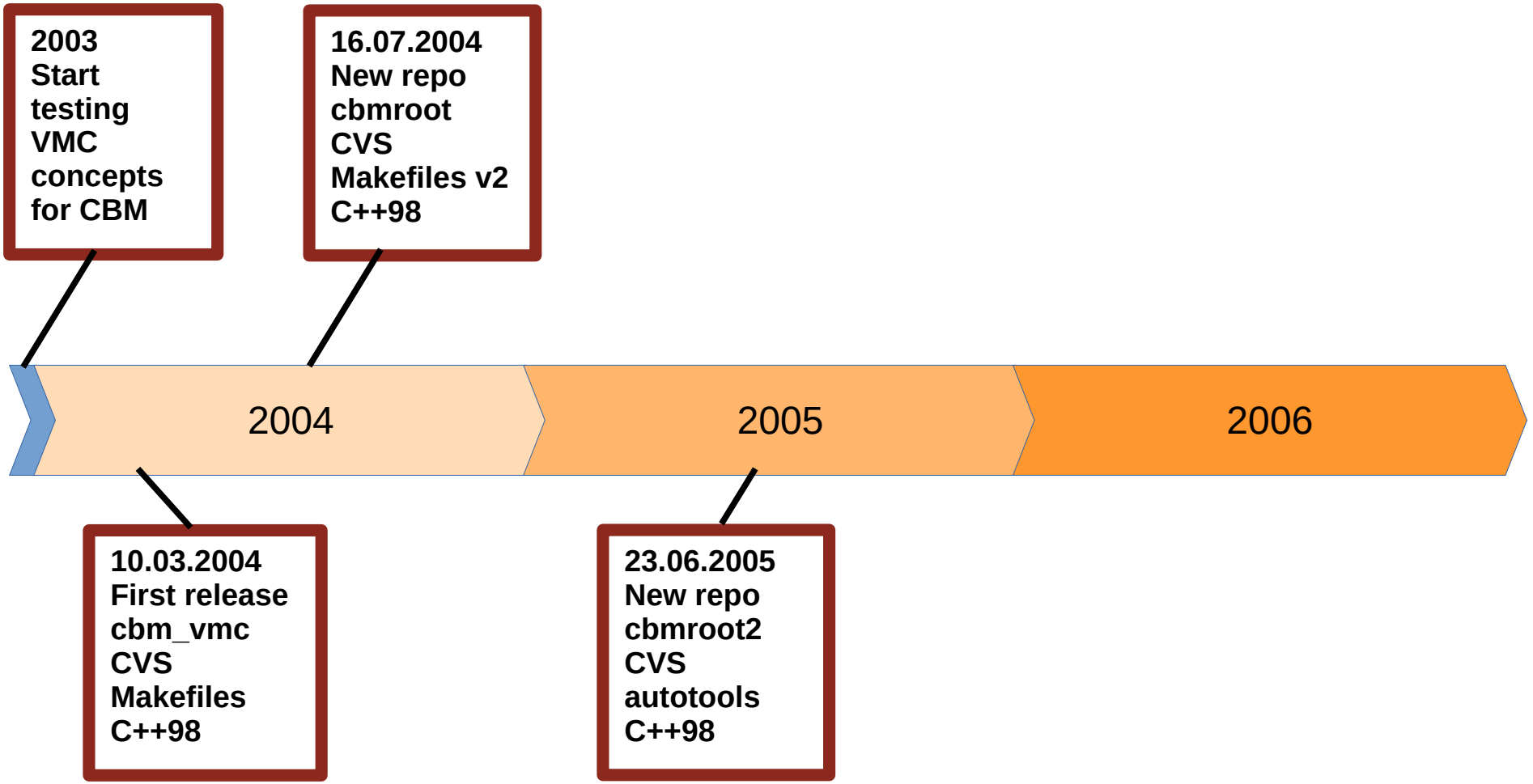
Compressed History



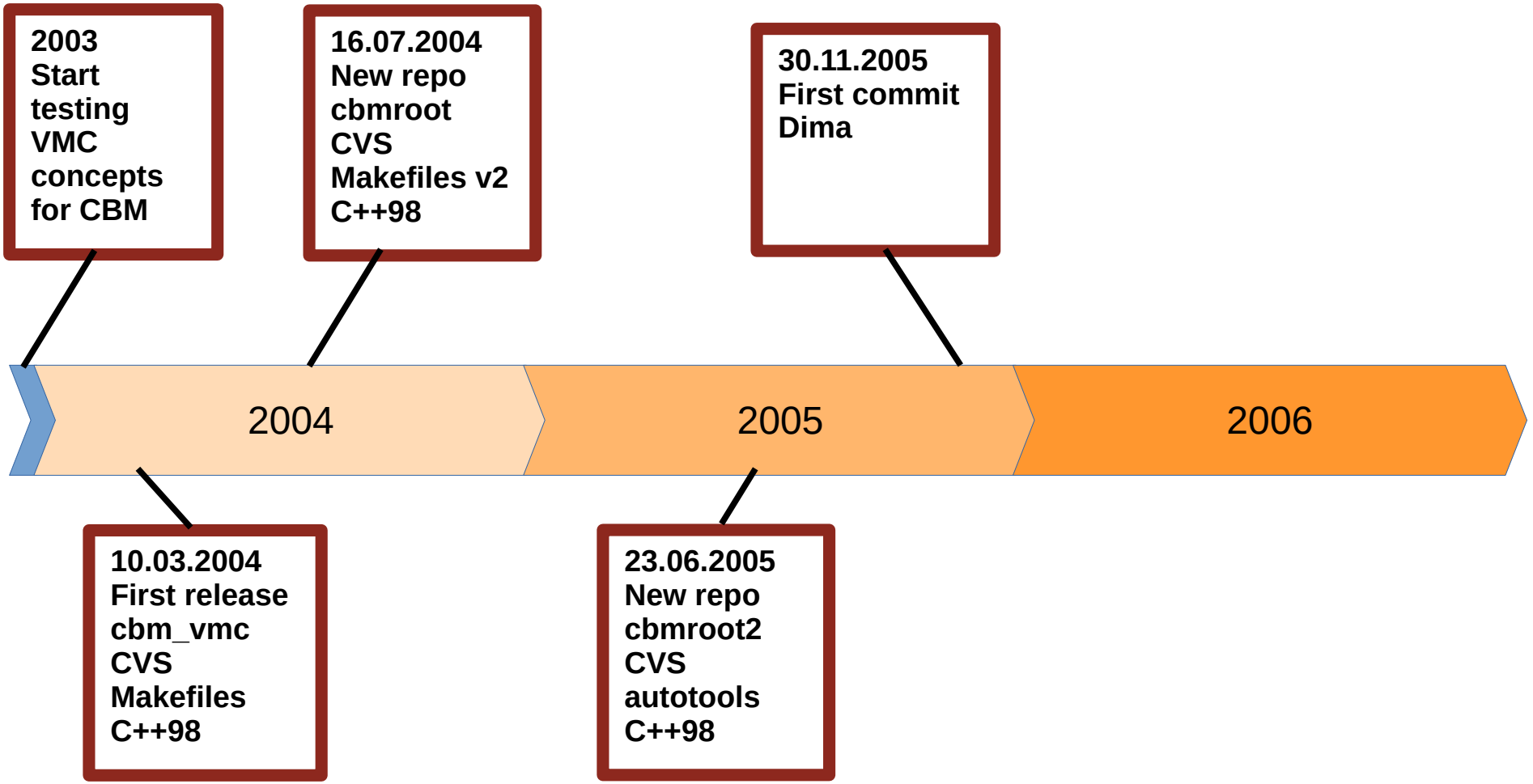
Compressed History



Compressed History



Compressed History



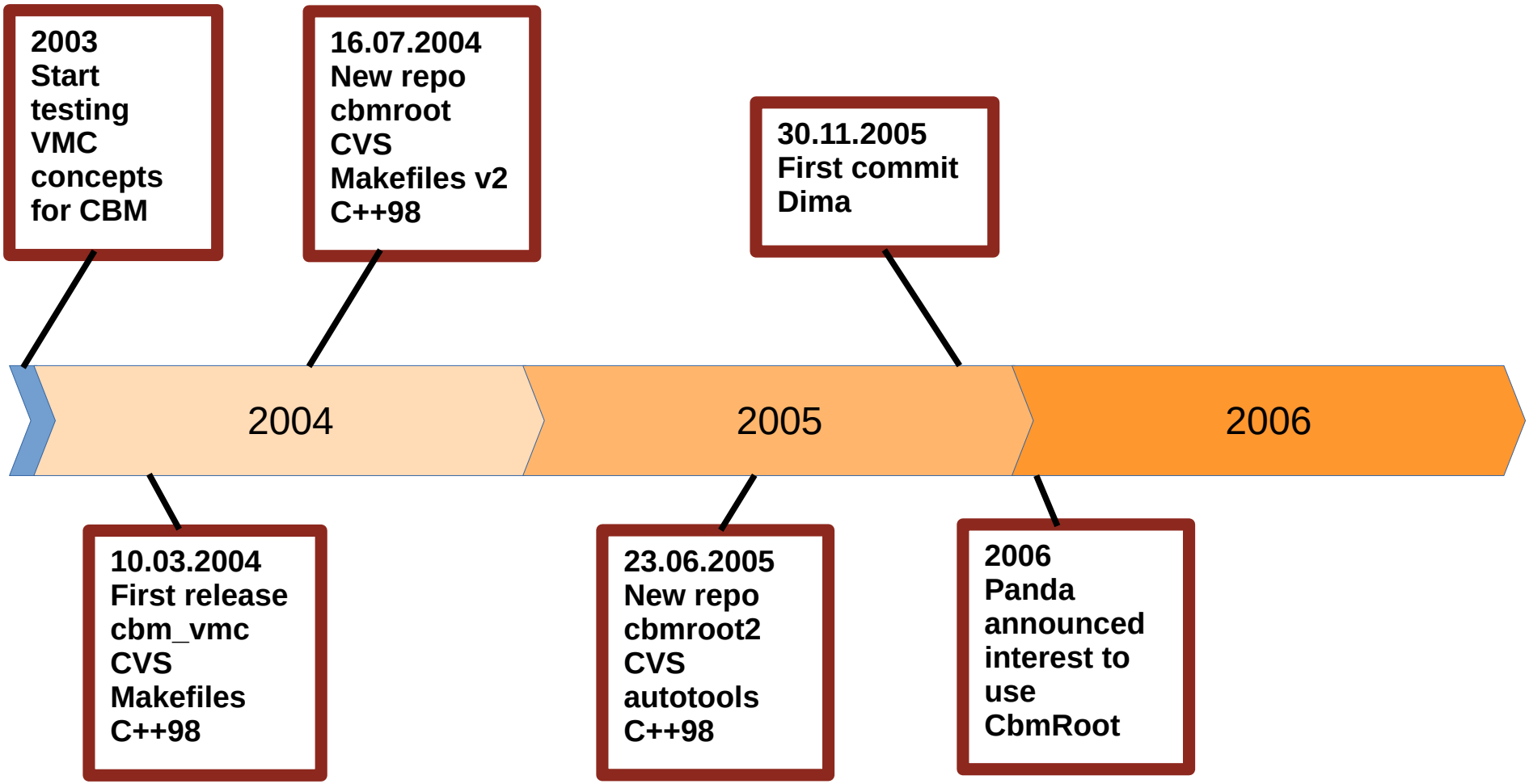
Compressed History

30.11.2005
First commit
Dima

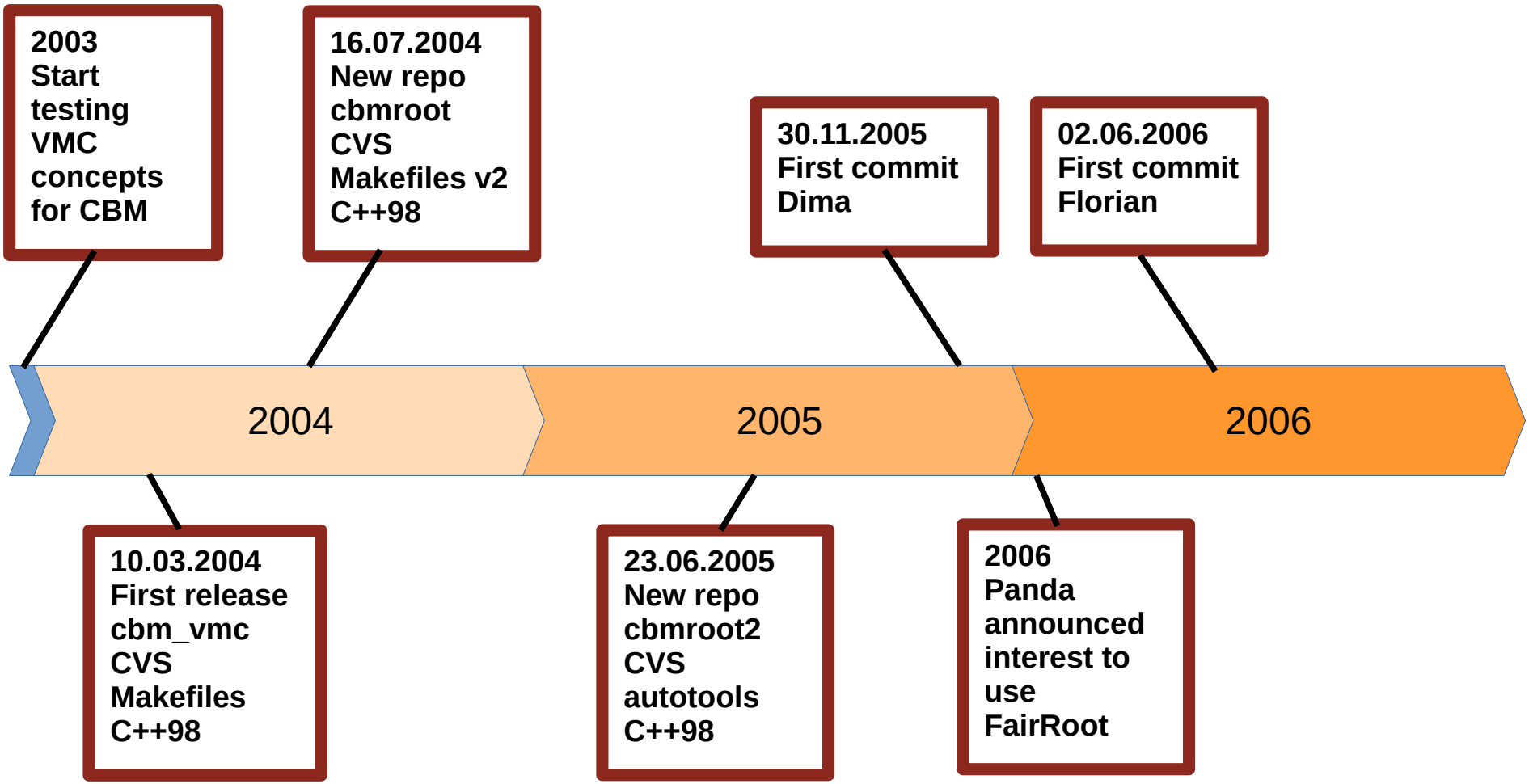
2005

2006

Compressed History



Compressed History



Compressed History

2003
Start testing VMC concepts for CBM

02.06.2006
First commit Florian

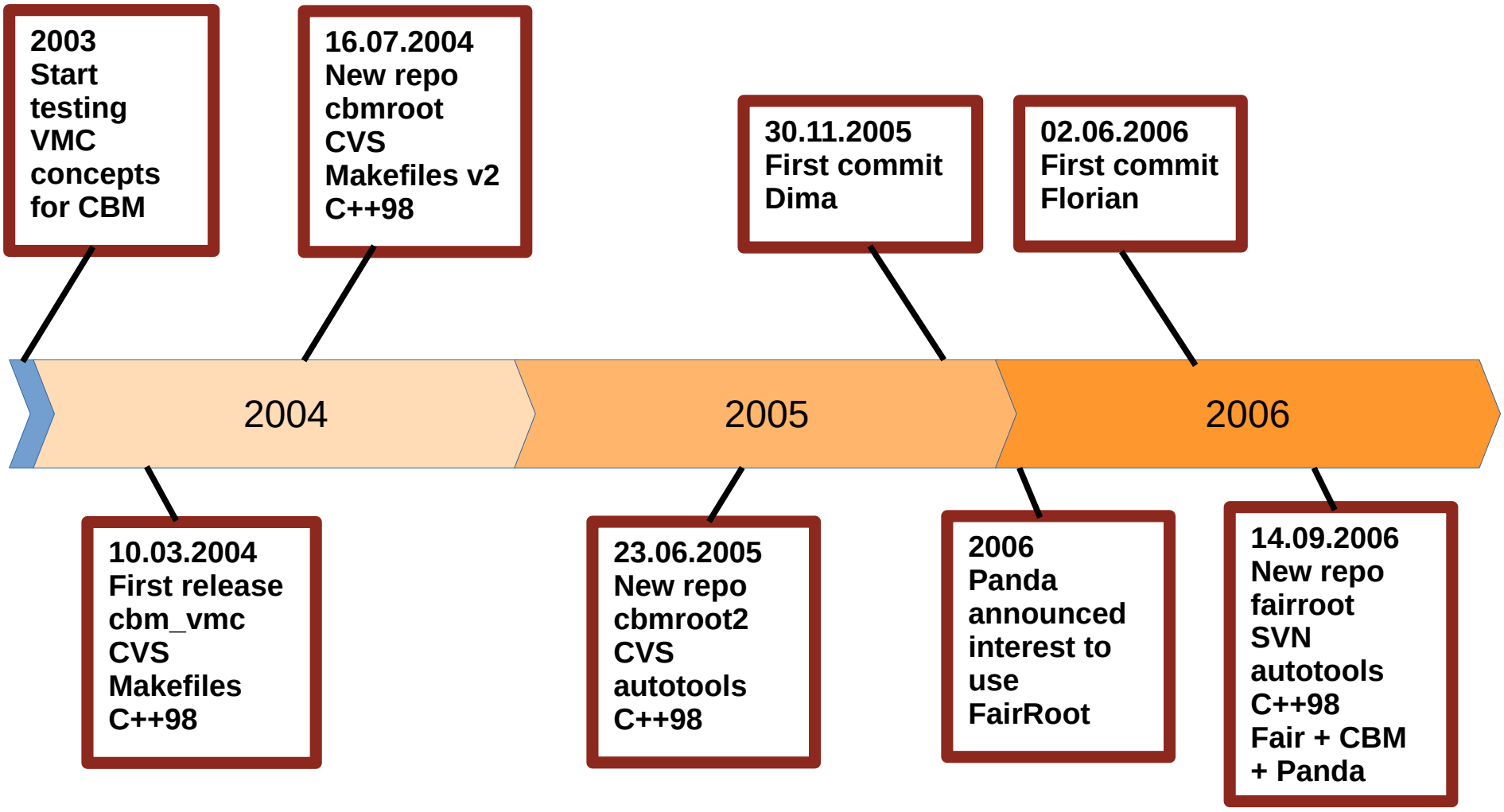
2004

2006

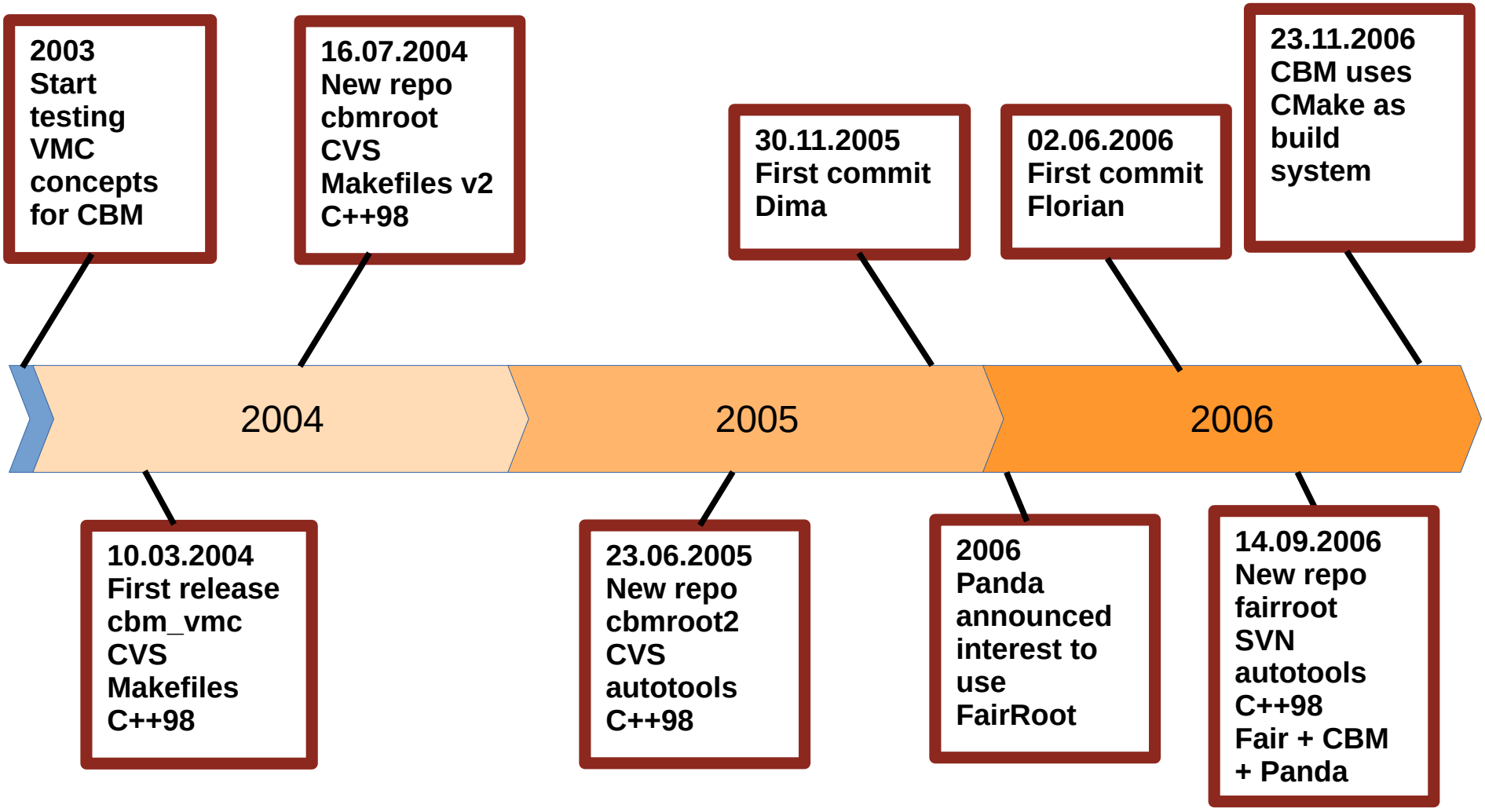
10.03.2004
First release cbm_vmc
CVS
Makefiles
C++98

2006
Panda announced interest to use FairRoot

Compressed History



Compressed History



Infrastrastructure

- Build system
  - Makefiles v1: 10.03.2004 – 16.07.2004
  - Makefiles v2: 16.07.2004 – 23.06.2005
  - Autotools: 23.06.2005 – 27.01.2009
  - CMake: 23.11.2006 - today
- C++ standards
  - C++98 till 2012
  - C++11 2012 – 2018
  - C++17 2018 – today

Repositories

- Repositories
  - 3 different CVS repos
    - cbm_vmc: 10.03.2004 – 16.07.2004
    - cbmroot: 16.07.2004 – 23.06.2005
    - cbmroot2: 23.06.2005 – 14.09.2006
  - SVN repository
    - fairroot: 14.09.2006 – 02.06.2020
    - Repository was restructured several times
    - Repository includes also the code of FairRoot and other experiments
  - GIT repository
    - CbmRoot: 03.06.2020 – today

Tools

- Which tools for software development do you know?

Tools

- Which tools for software development do you know?
  - Editor / IDE
  - Compiler / Interpreter (C++ / Python)
  - Build and Test system
  - Debugger
  - Source code management
  - Issue Tracker
  - Forum
  - Continuous Integration
  - Dashboard for CI results
  - Container runtimes
  - ...

Tools

- Which tools for software development do you know?
  - Editor / IDE
  - Compiler / Interpreter (C++ / Python)
  - Build and Test system
  - Debugger
  - Source code management
  - Issue Tracker
  - Forum
  - Continuous Integration
  - Dashboard for CI results
  - ...
- Which of these tools do you use?

Tools used for CbmRoot development

- Source code management / Version control system: **git**
- Web frontend to source code management: **GitLab**
- Build and Test system: **CMake**
- Continuous Integration: **Gitlab and own infrastructure**
- Dashboard for CI results: **CDash**
- Issue Tracker: **Redmine**
- Container Runtimes: **Docker and Apptainer**
- Code Formatting: **clang-format**

Tools used in CBM context

- Wiki: **FosWiki**
- Mailing Lists: **ListServ**
- Collaboration Database: **Own development**
- Webpage: **Drupal**
- Video Conferences: **Zoom**
- Meeting Management: **Indico**

CMake

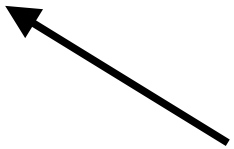
- CMake is an open source meta build system
- CMake generates native build environment
  - Makefiles
  - IDE project files
  - ...
- CMake is a set of tools
  - cmake – build environment generator
  - ctest – tool for automatic testing
  - cpack – packaging tool
  - ccmake – graphical frontend to cmake
- CMake is operating system and compiler independent

CMake

- CMake is controlled by creating CMakeLists.txt files in each directory which belongs to a project
- CMake creates a very accurate dependency tree
  - A rerun of the build process compiles only files which have changed and all their dependencies. All other files are not recompiled
- Most of our CMakeLists.txt only define variables and finally call one command
  - Example on the next slides
- Developer normally needs to add new files to a variable
  - For anything else or in case of problems please contact the software team

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )


set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```

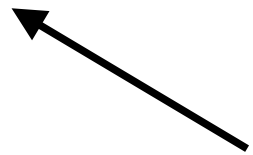


Directories with needed headers

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )


set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```



List of source files

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )


set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```

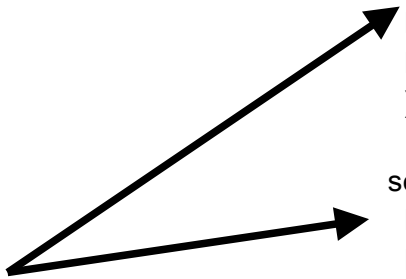
Library name
Linkdef file name

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )


set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```



Library dependencies target not lib names
Private: only used in source file
Public: used in headers

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )


set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```

Here the magic happens

Example CMakeLists.txt

```
set(INCLUDE_DIRECTORIES
  ${CMAKE_CURRENT_SOURCE_DIR}
  )
```

Directories with needed headers

```
set(SRCS
  CbmFieldConst.cxx
  CbmFieldContFact.cxx
  CbmFieldMap.cxx
  CbmFieldMapCreator.cxx
  CbmFieldMapData.cxx
  CbmFieldMapSym2.cxx
  CbmFieldMapSym3.cxx
  CbmFieldPar.cxx
  CbmBsField.cxx
  CbmFieldCreator.cxx
  CbmFieldMapDistorted.cxx
  CbmFieldMapSym1.cxx
  )
```

List of source files

Library name
Linkdef file name

```
set(LIBRARY_NAME CbmField)
set(LINKDEF ${LIBRARY_NAME}LinkDef.h)
set(PUBLIC_DEPENDENCIES
  FairRoot::Base
  FairRoot::ParBase
  ROOT::Core
  )

set(PRIVATE_DEPENDENCIES
  ROOT::MathCore
  ROOT::RIO
  ROOT::Hist
  ${VMCLIB}
  )

generate_cbm_library()
```

Library dependencies
target not lib names
Private: only used in source file
Public: used in headers

Here the magic happens

CTest

- Ideal integration with CMake
- Easy to setup new tests
- A test can be anything which is executable
  - Scripts (shell, perl ..)
  - Executable
  - Root macros
- Help to automatically identify problems when they occur
- Direct feedback for developers when they play with new features
- CTest is used for our CI infrastructure

Tests

- Tests are defined also in CMakeLists.txt files
- Find many examples in macro directory
- Executing a ROOT macro is as easy as shown in macro/field/CMakeListst.txt
  - Generate shell script to setup the runtime environment and execute the ROOT macro
  - Adds the tests to the list of existing tests
  - Set test properties (max. runtime, and successful string)

```
GENERATE_ROOT_TEST_SCRIPT(${CBMROOT_SOURCE_DIR}/macro/field/FieldMapTest.C)

add_test(field_field ${CBMROOT_BINARY_DIR}/macro/field/FieldMapTest.sh)

SET_TESTS_PROPERTIES(field_field PROPERTIES TIMEOUT "60")
SET_TESTS_PROPERTIES(field_field PROPERTIES PASS_REGULAR_EXPRESSION "Test Passed;All
ok")
```

How to contribute to a software project?

- How to get new files or file changes into the code base of a project?
- Typical project use cases
  - Single developer
    - Can do whatever he wants
  - Small number of developers
    - Only few or many changes
    - Developers working on same or different code parts
    - Probably coordination needed to avoid interference
  - Large number of developers
    - Very likely the developers will interfere with each other
    - Without coordination the development becomes impossible

Workflow

- How to coordinate your work?
  - Everybody can add/change files in a common workplace
  - Developers can only add/change files in subparts of the workplace
  - Only single person can add/change files
    - Send changes using mail
    - Let him copy your code (if possible)
  - Use source code management
    - Several tools available (CVS, SVN, git, …)
    - Some of them already define a workflow
    - Has several advantages
- Define a workflow as early as possible
- Use the workflow

Workflow for CbmRoot development

- Over the years we used several workflows
  - Depending on the source code management system
  - Beginning: everybody could do everything
  - Intermediate: developer had permissions on directory/file level
  - Now: again everybody can do everything BUT
    - changes are reviewed automatically
    - If automatic review was successful another developer reviews the changes
    - Only if both steps were successful the changes are added to the code base

Source Code Management
Underlying problem

- How do you keep track of changes made to your files?
  - Do the bookkeeping yourself.

    Create new files every time you do a change.
    - Will create a huge number of files which are complicated to organize
    - May blow up your disk space
  - No metadata information if not saved separately
- How do you communicate changes to your collaborators?
  - Send changes by mail?
  - Work all in the same account?
- Use a system which hides all the complications from the user

Version control

- A version control system (VCS) manages documents over time
  - A VCS keeps the history of all changes
  - Many versions of every file
  - Allows to go back to an older version of the file
  - Show differences between different versions
  - Log messages name the reason for changes
  - ...
- A VCS coordinates the work of multiple authors
  - Avoid conflicts between the developments of different developers
- A VCS allows user authentication and controlled access to files
  - Read/Write permissions for user and groups to files and directories

Which information is stored?

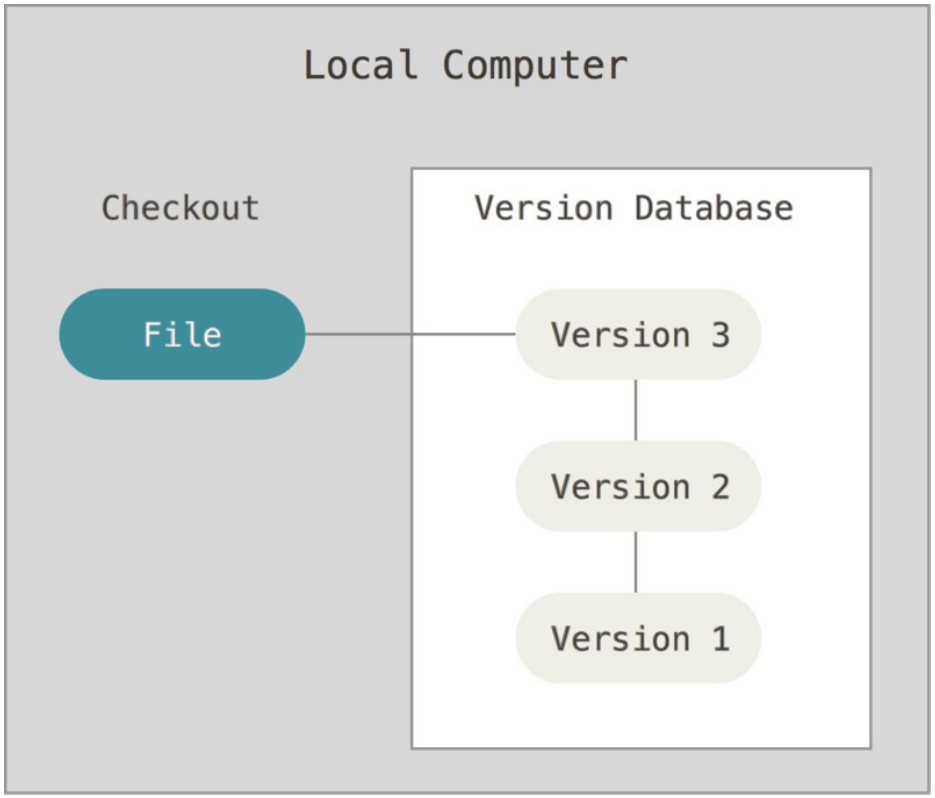
Content	what has changed?
Date	when did it change?
Author	who changed it?
Reason	why has it changed?

VCS does this

you enter this

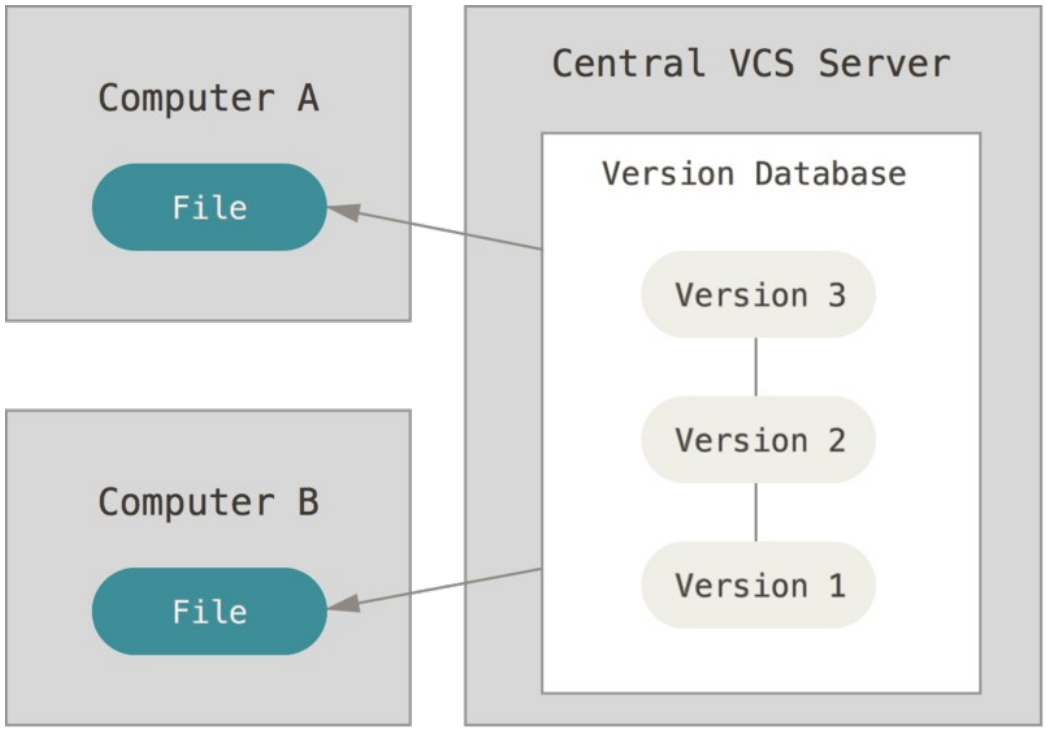
- The Reason should be meaningful and explanatory !!
- Don’t say how you have changed but why and what
  - Bad: bug-fix
  - Good: Correctly init variable x because of division by zero
- Separate subject from body with a blank line
- Limit the subject line to 50 characters
- Wrap the body at 72 characters

Local Version Control System



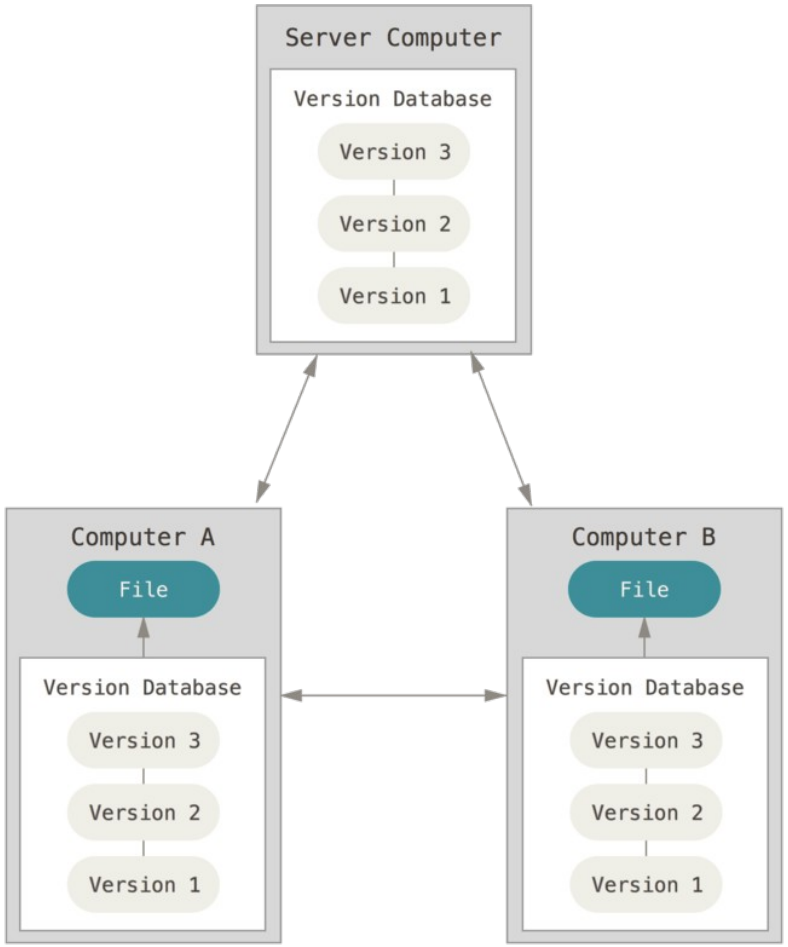
- Store differences between versions (deltas) for each file in a special format
- Less error prone than manually managing file versions
- E.g. RCS

Central Version Control System



- Allows a collaboration of different developers
- Fine grained access control is possible
- Central Server is a single point of failure
- Commits went always to the central server
- If the server is locally on your computer it works as local VCS
- E.g. CVS or Subversion

Distributed Version Control System



- Commits are done first locally which allows to save the full history of changes locally
  - Each computer has a copy of the full history
  - No single point of failure since every client copy is a full backup
- Depending on workflow synchronization can be complicated
  - Most workflows use a central server
- E.g. Git, Mercurial

Workflow again

- Local and central version control systems define a more or less fixed workflow
- With a distributed version control system many different workflows become possible
- Git is a very powerful toolbox to implement many different workflows
  - Good: very powerful and flexible
  - Bad: very powerful and flexible
- For CbmRoot we chose to use GitLab as repository
  - A software repository, or repo for short, is a storage location for software packages (Wikipedia)

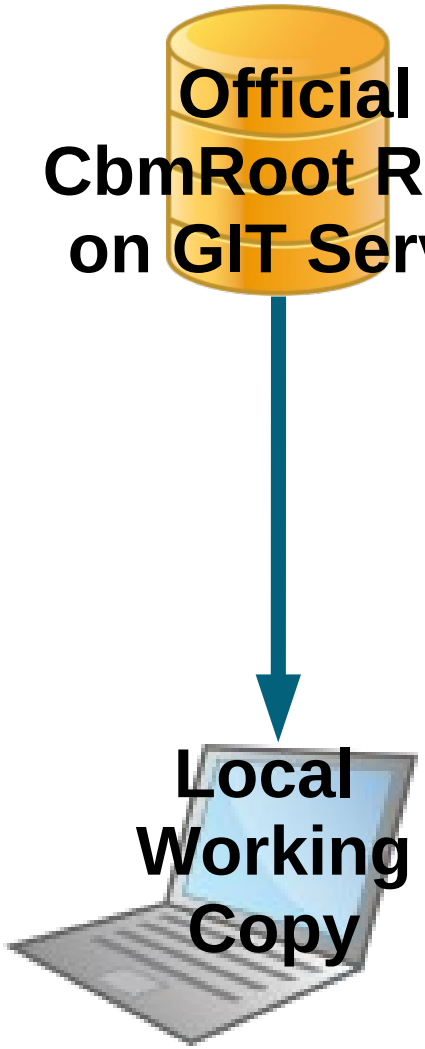
Repository

- The “Central Repository” is the official source of the project on a central server
  - GitHub or GitLab are well known repository providers
  - Allow to implement workflows
  - Everything else isn’t an official version !!!
- A “Fork” is a private clone (copy) of the official repository on the central server at a given point in time
  - Done using the services provided by GitHub or GitLab
  - Allow to do changes without affecting the central repository
  - Not synchronized automatically with official repository
  - Needed to integrate changes into the “Central Repository”
- The “local repository” or “working copy” is a clone from the central server on your local computer

Naming convention

- User
  - Person who only intends to download and use the CbmRoot source code
- Developer
  - Person who intends to get code into the official CbmRoot repository
- Manager
  - Person who manages the official CbmRoot repository
    - 5-6 dedicated people
    - https://git.cbm.gsi.de/computing/cbmroot/-/project_members
  - This workflow is not covered here

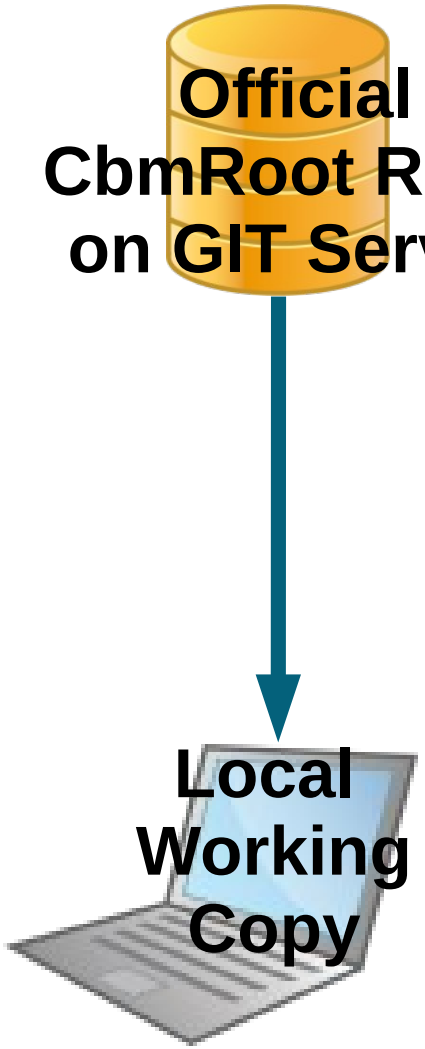
Workflow for users



Get the code initially

git clone https://git.cbm.gsi.de/computing/cbmroot

Workflow for users



Update the code

```
cd <source_directory>
git fetch origin
git rebase origin/master
```

Summary for users

- Very simple workflow
- Either get the code initially or update your local working copy to the latest state

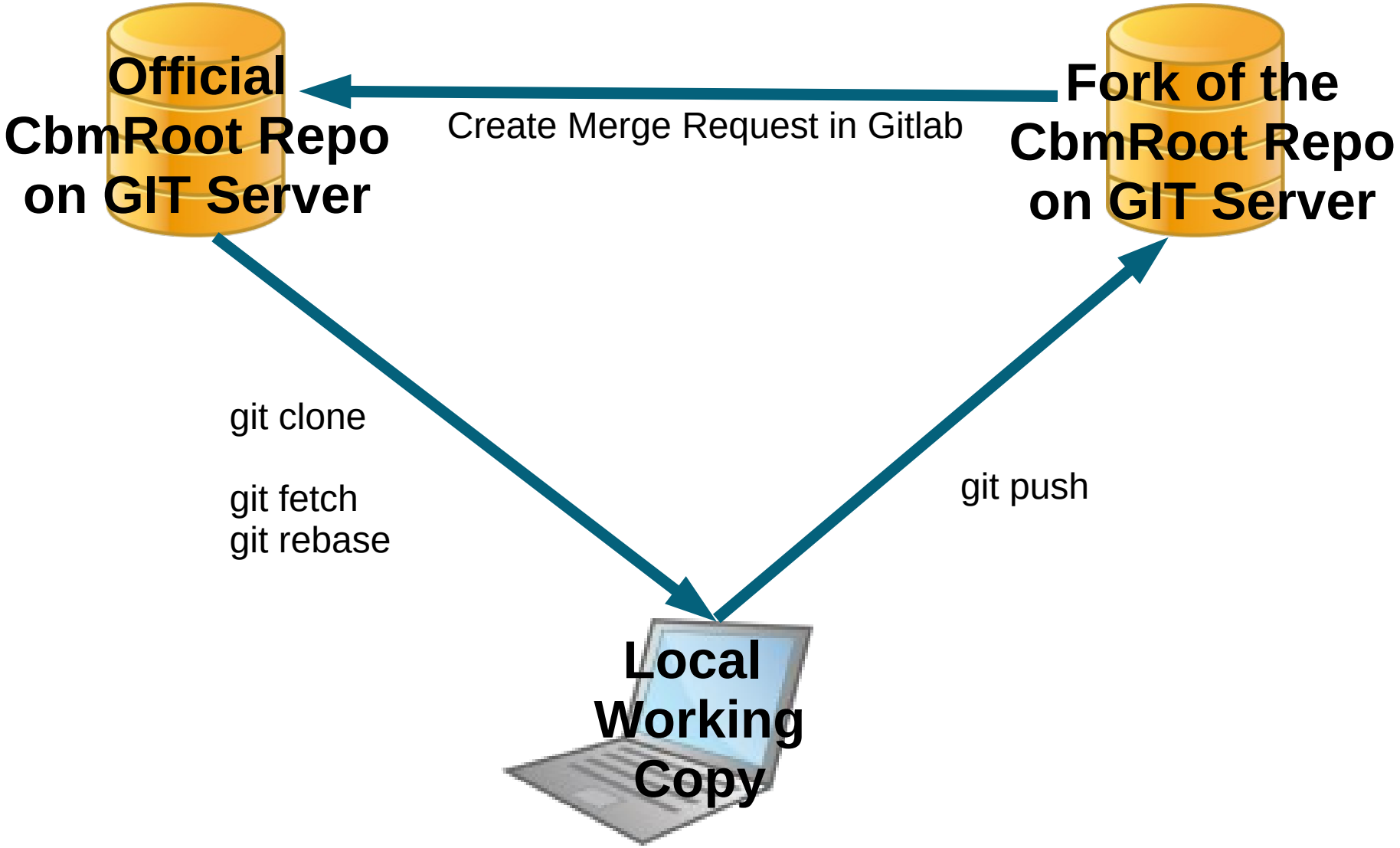
Workflow for developers

- Developers get the code in the same way as the users
- Developers get updates in the same way as the users
- Nobody is allowed to upload (push) changes to the “Official directory”
  - This is even not allowed for administrators !!
- How do a developer get his local changes back into the “Official Repository”?
  - If the direct way is blocked we have to use a bypass
  - The following slides only cover this „Merge Request“ workflow

Merge Request Workflow

- The developer offers his code and asks for adding it
  - Like sending a mail with patches to the main developer asking him to add them to the official code
- Workflow is completely implemented in GitLab
  - Most steps are automatic
  - Relevant people are informed via mails
  - Allow extensive testing and peer review before adding code
- Advantages for Maintainers
  - Reduce workload enormously
    - Problems are mostly found before adding the code
    - In case of problems the developer has to fix them

Workflow for developers
Final Picture



Workflow in Detail

- Have you seen this workflow already?
- Have you used the workflow before?
- The workflow is described in the following presentation
  - CbmRoot Merge Request Workflow

Actions after Merge Request

- GitLab creates a new invisible branch in the main repository
  - The branch is deleted after the code is merged
  - Get the code in the local repository
    - git fetch <repo_name> merge-requests/<id>/head:<local_branch_name>
    - e.g. git fetch upstream merge-requests/1879/head:mr-upstream-1879
- GitLab starts the Continuous Integration
  - All stages and tasks of the CI are defined the file .gitlab-ci.yml
  - GitLab does only the orchestration, the actual work is done on a dedicated CI infrastructure
  - If the full CI chain was successful the Merge Request is marked to be merged
  - GitLab send mails to the Code Owners to inform them about the MR

Continuous Integration

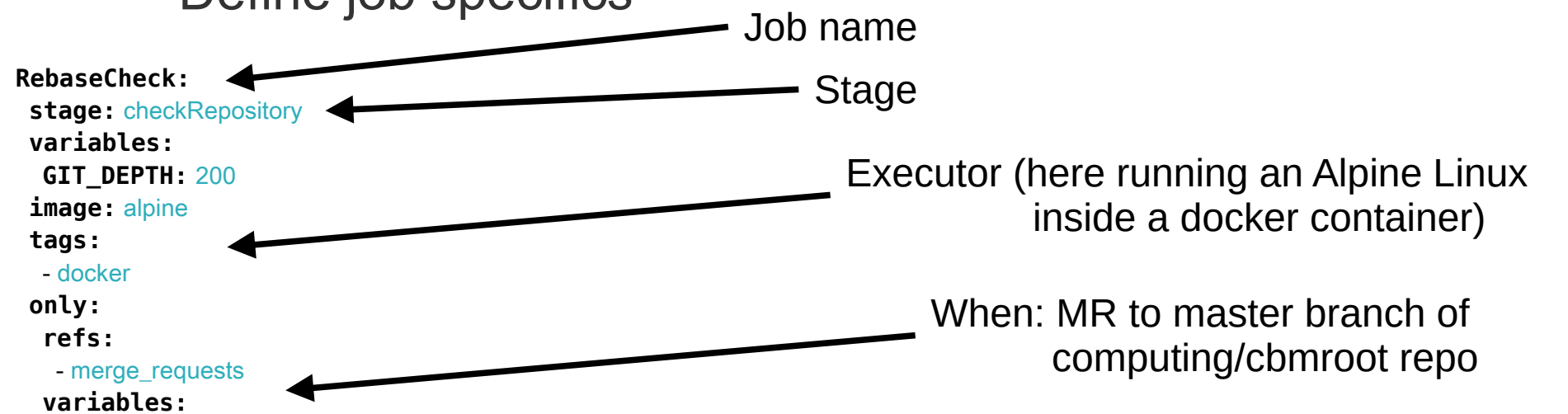
- GitLab start jobs on the CI infrastructure as defined in the CI configuration
  - Jobs in stages can run in parallel
  - The stages run sequentially
- GitLab runners on the CI build nodes wait for work
  - Pull jobs from GitLab server
  - One runner can execute several executors
- Different Executors runners for different tasks
  - Docker Executor to do work inside docker containers
  - Apptainer Executor to build and test CbmRoot with the GSI specific VAEs (Debian 10, Debian 11 and CentOS 7)
  - Shell Executor to build and test CBmRoot on macosx

CI config

- Define Test Stages
  - Once at the beginning of the config
  - Several jobs per Stage possible

```
stages:
 - checkRepository
 - checkFormat
 - build
 - package
 - verify
 - finalise
 - documentation
```

- Define job specifics




```
RebaseCheck:
 stage: checkRepository
 variables:
  GIT_DEPTH: 200
 image: alpine
 tags:
  - docker
 only:
  refs:
   - merge_requests
  variables:
   - $CI_MERGE_REQUEST_PROJECT_PATH == "computing/cbmroot" && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "master"
```

CI config II

- Code
  - Executable or script
  - Need to return 0 on success

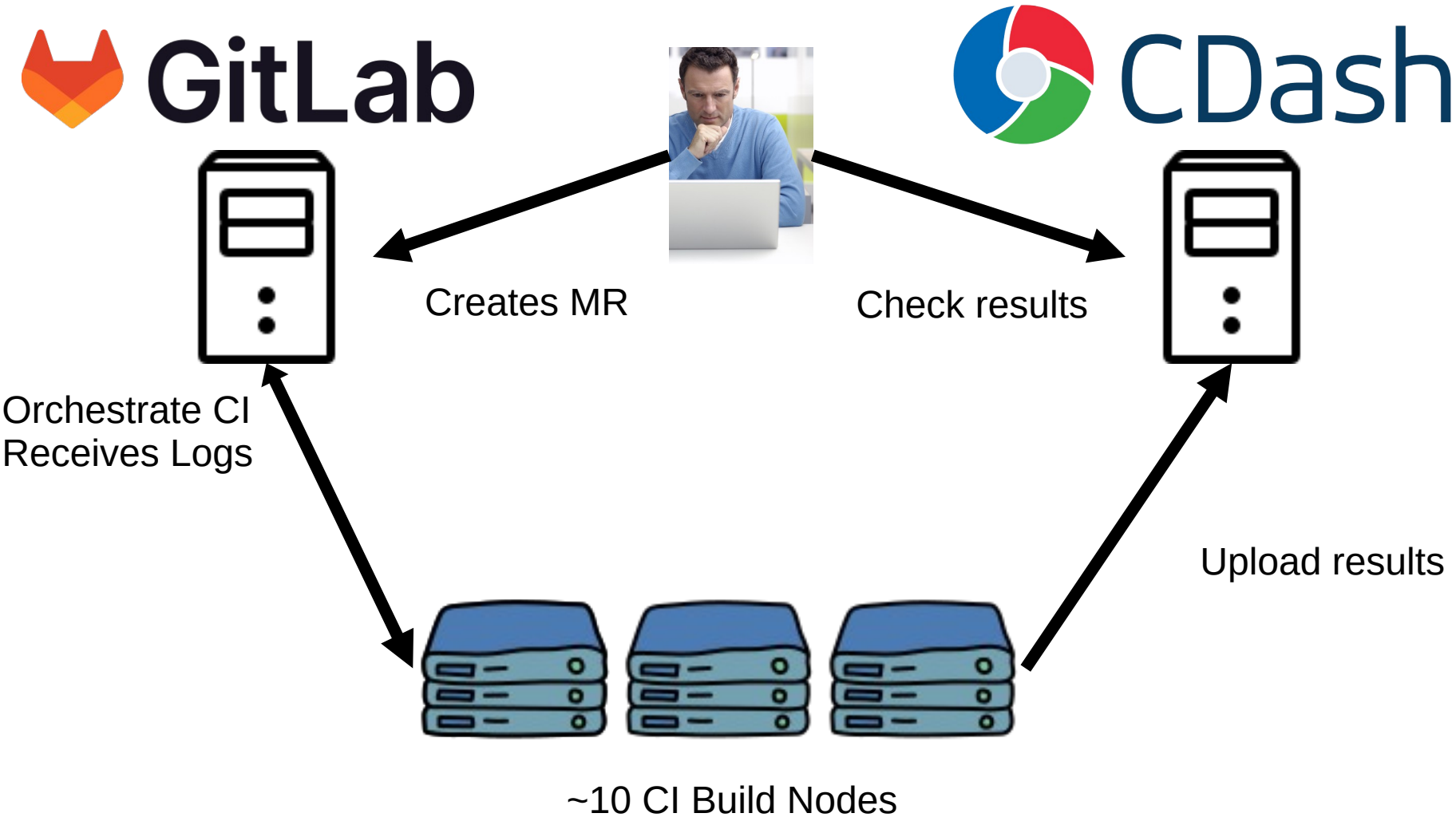
```
script:
 # Get the upstream repository manually. I did not find any other way to have it for
 # comparison
 # Check if a rebase is needed
 # If a rebase is needed stop immediately
 - apk update && apk add git bash
 - scripts/connect_upstream_repo.sh $CI_MERGE_REQUEST_PROJECT_URL
 - git fetch upstream
 - hash1=$(git show-ref upstream/$CI_MERGE_REQUEST_TARGET_BRANCH_NAME | cut -f1 -d' ')
 - hash2=$(git merge-base upstream/$CI_MERGE_REQUEST_TARGET_BRANCH_NAME HEAD)
 - echo "${hash1}"
 - echo "${hash2}"
 - if [ "${hash1}" = "${hash2}" ]; then
 -   echo "No rebase required"
 -   exit 0
 - else
 -   echo "The Merge Request is not up-to-date"
 -   echo "Rebase is required"
 -   exit 1
 - fi
```

Update Alpine

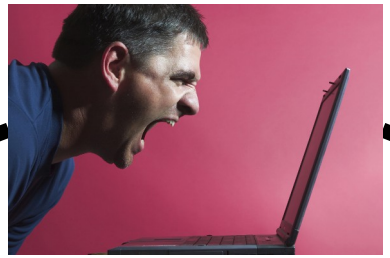
Connect upstream reository

Check if HEAD of upstream repo is in the history

CI Infrastructure



CI Infrastructure



Creates MR

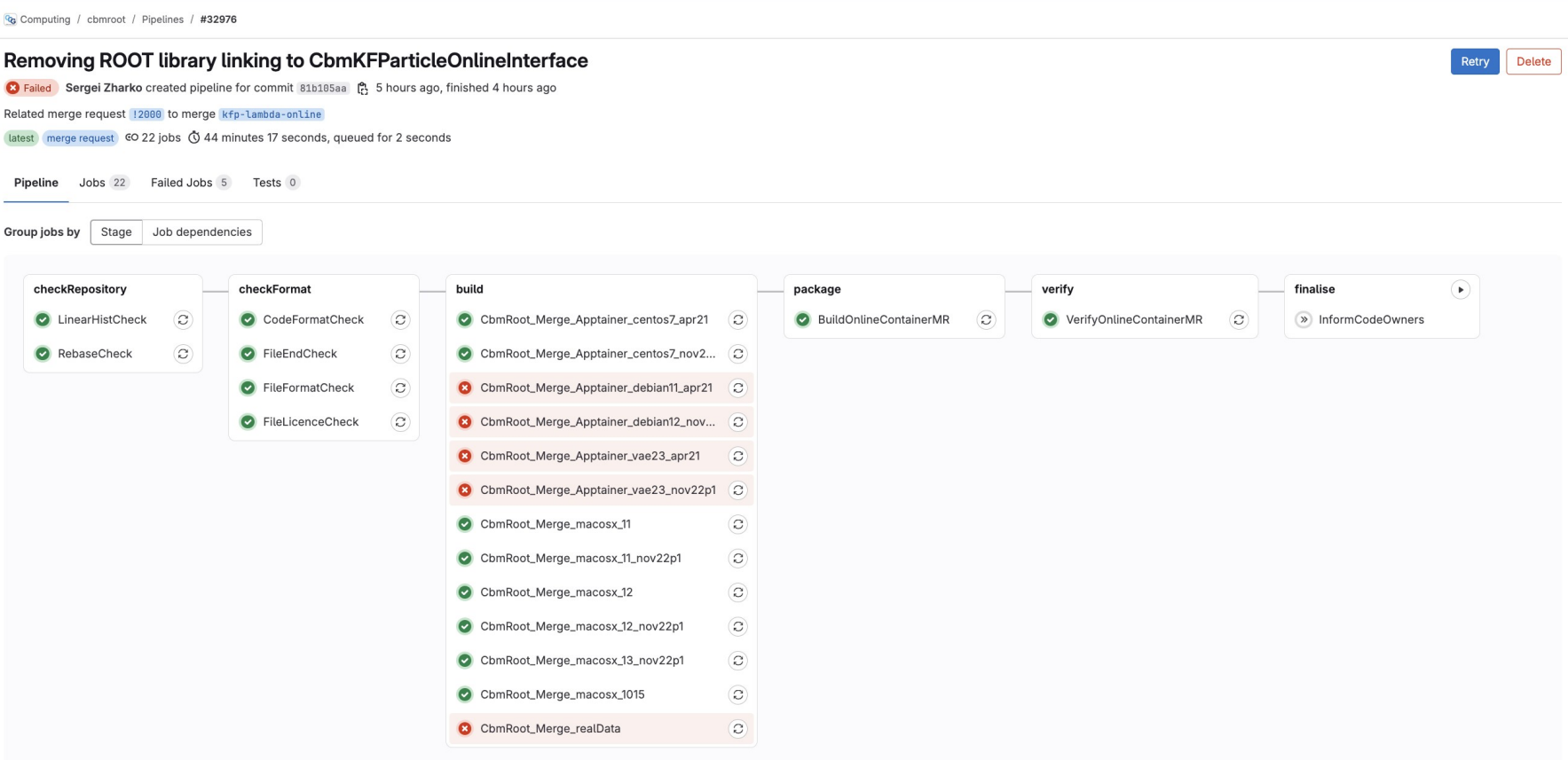
Check results

Orchestrate CI
Receives Logs

Upload results

~10 CI Build Nodes

CI Pipeline



- Clicking on one of the jobs will open the detailed log output of that job

CI Job Log

```
195/195 Test  #36: load_libraries ...................................   Passed  860.75 sec
100% tests passed, 0 tests failed out of 195
Total Test time (real) = 1049.03 sec
Submit files
   SubmitURL: https://cdash.gsi.de/submit.php?project=CbmRoot
   Uploaded: /tmp/custom-executor215560493/builds/computing/cbmroot/build/Testing/20250215-1040/Test.xml
   Submission successful
Performing coverage
 Cannot find any coverage files. Ignoring Coverage request.
Submit files
   SubmitURL: https://cdash.gsi.de/submit.php?project=CbmRoot
   Submission successful
_ctest_test_ret_val: 0
_install_ret_value: false
--
--  You can find the produced results on the CDash server
--
--  CDash Build Summary ..: https://cdash.gsi.de/buildSummary.php?buildid=494149
--  CDash Test List ......: https://cdash.gsi.de/viewTest.php?buildid=494149
--
Running after_script                                                    00:08
Running after script...
$ rm -rf build
Cleaning up project directory and file based variables                  00:07
Job succeeded
```

- Log output helps to find problems
- At the end of some of the jobs information is also uploaded to the CDash server
  - Direct links are at the end of the log information
  - Same information can also be access from CDash

CDash Server

- CDash aggregates, analyses and displays the results of software testing processes
  - https://cdash.gsi.de
  - CbmRoot: https://cdash.gsi.de/index.php?project=CbmRoot
- Clients can be located anywhere world wide
  - Everybody can upload information
  - Users can provide information for systems not yet in the test matrix
- Find problems on systems you don’t have access
  - For most CbmRoot users this are macosx systems
- Allows to see subtle changes over time
  - e.g. increasing test time after some code change

CDash and CMake

- CDash is very well integrated with CMake
  - No extra work to upload build and test results
- Our CMake configuration define several build targets
  - Merge Request (Is run for merge request branches)
  - Nightly (Is used for extended test suite, normally once per day)
  - Experimental (Can be used to upload local info to CDash)
  - ...
- Experimental test
  - Does not update the state of the local CbmRoot source code
  - Does not delete the build directory
  - Build CbmRoot, runs the test suite and upload the results to Cdash
  - Useful to share local problems with the CbmRoot community

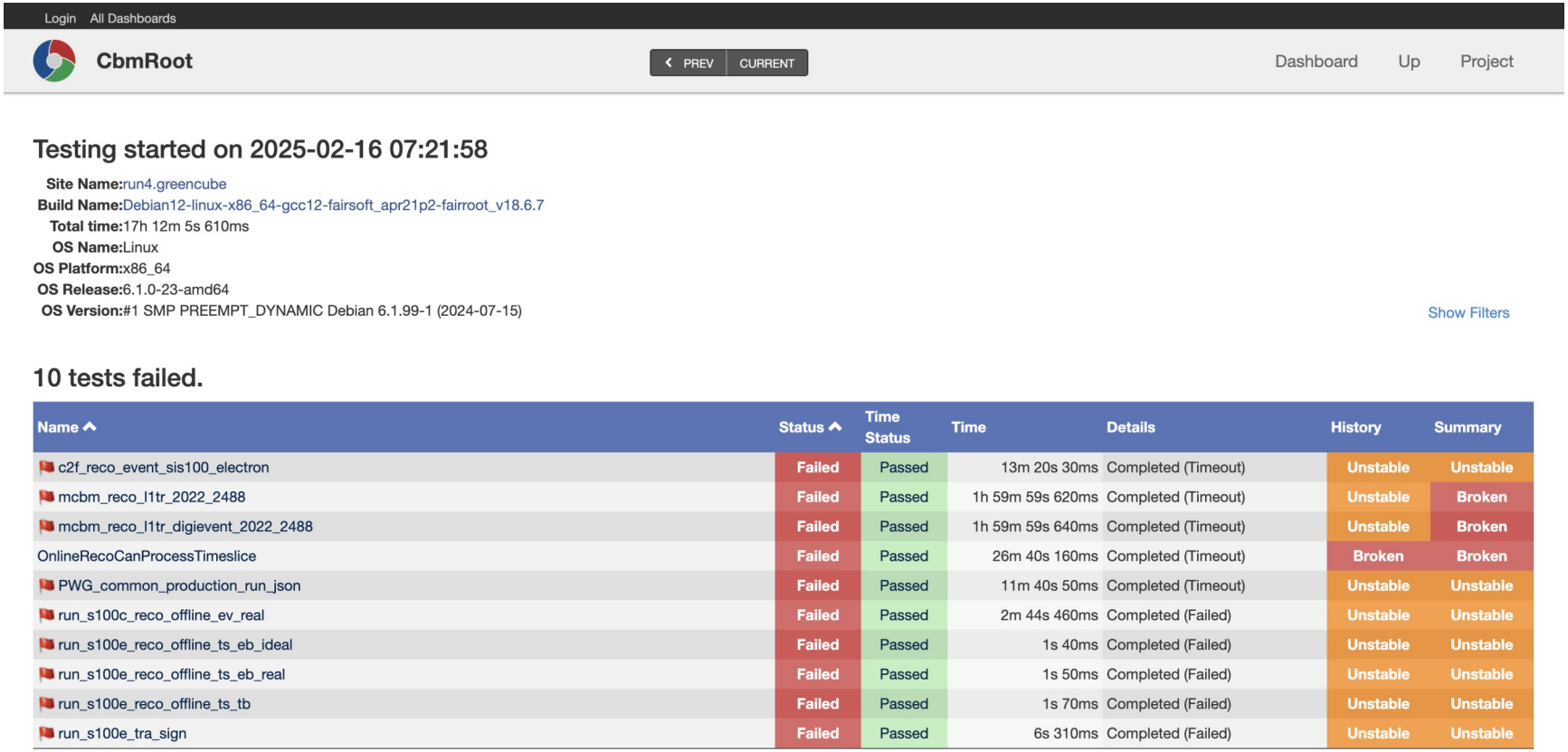
CDash Main Page

Nightly *3 builds* [view timeline]

		Update	Configure		Build		Test					
Site	Build Name	Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Time	Start Time	Labels
run4.greencube	Debian12-linux-x86_64-gcc12-fairsoft_nov22p1-fairroot_v18.8.0	3614fb	0	2	0	32^{+6}_{-6}	3^{+1}	5^{+4}_{-1}	207_{-4}	3h 54m 27s	10 hours ago	(none)
run4.greencube	Debian12-linux-x86_64-gcc12-fairsoft_apr21p2-fairroot_v18.6.7	3614fb	0	2	0	31^{+6}_{-6}	10^{+8}_{-2}	10^{+9}_{-3}	195^{+4}_{-16}	1 3h 54m 34s	6 hours ago	(none)
mcbmop09.cbmc.gsi.de	Debian-12.9-linux-x86_64-gcc12-fairsoft_nov22p1-fairroot_v18.8.0	3614fb	0	1	0	15^{+6}_{-5}	4^{+4}	4^{+4}_{-1}	262_{-7}	2 56m 9s	3 hours ago	(none)

- Most of the elements are links to further and more detailed information
  - e.g. click on a number in the column “Test Fail”
- In the following slides there are some examples
- Try it out yourself

CDash Failed Tests



Login All Dashboards

CbmRoot

PREV CURRENT

Dashboard Up Project

Testing started on 2025-02-16 07:21:58

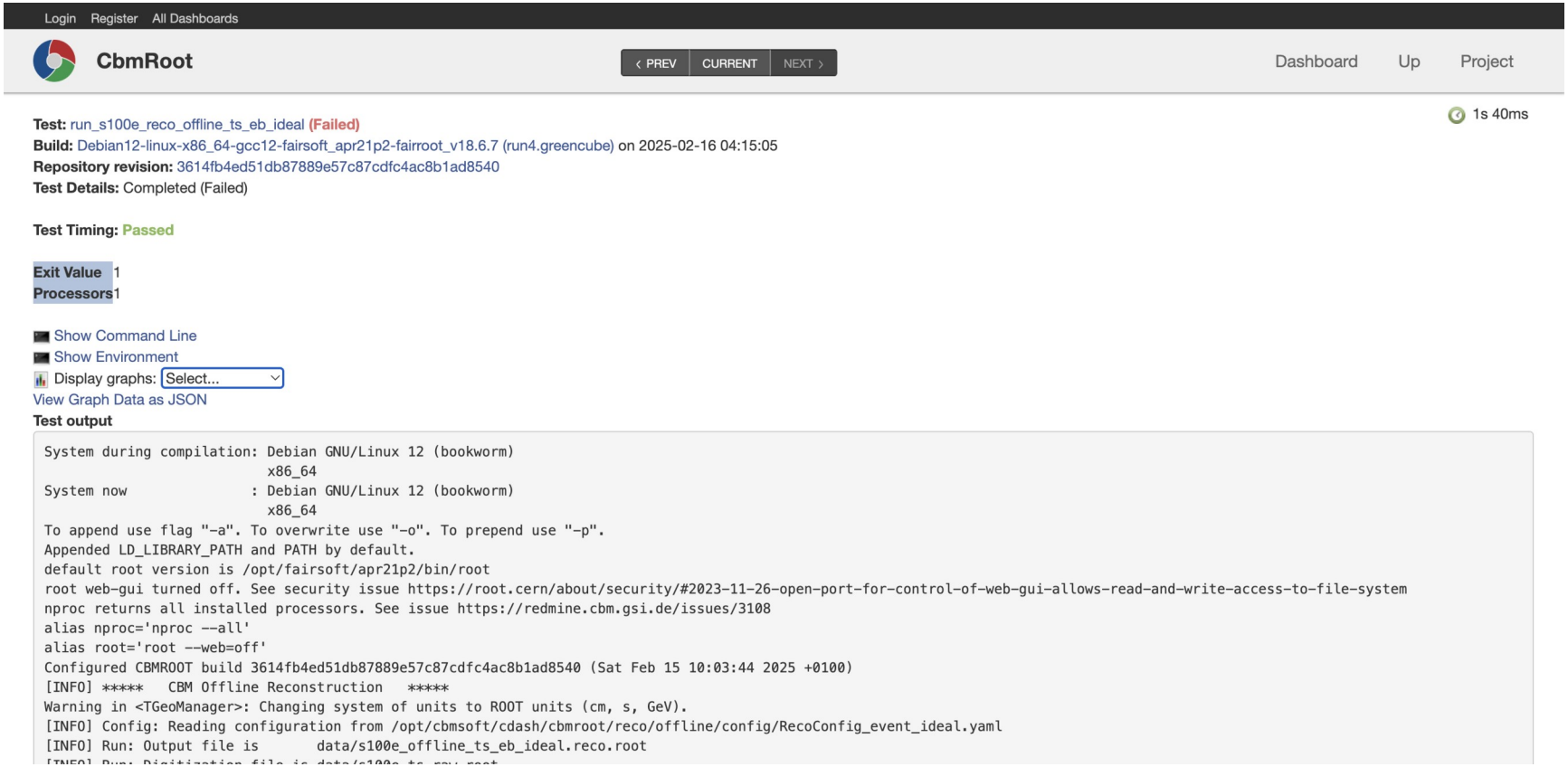
Site Name:run4.greencube
Build Name: Debian12-linux-x86_64-gcc12-fairsoft_apr21p2-fairroot_v18.6.7
Total time:17h 12m 5s 610ms
OS Name: Linux
OS Platform:x86_64
OS Release:6.1.0-23-amd64
OS Version:#1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15)

Show Filters

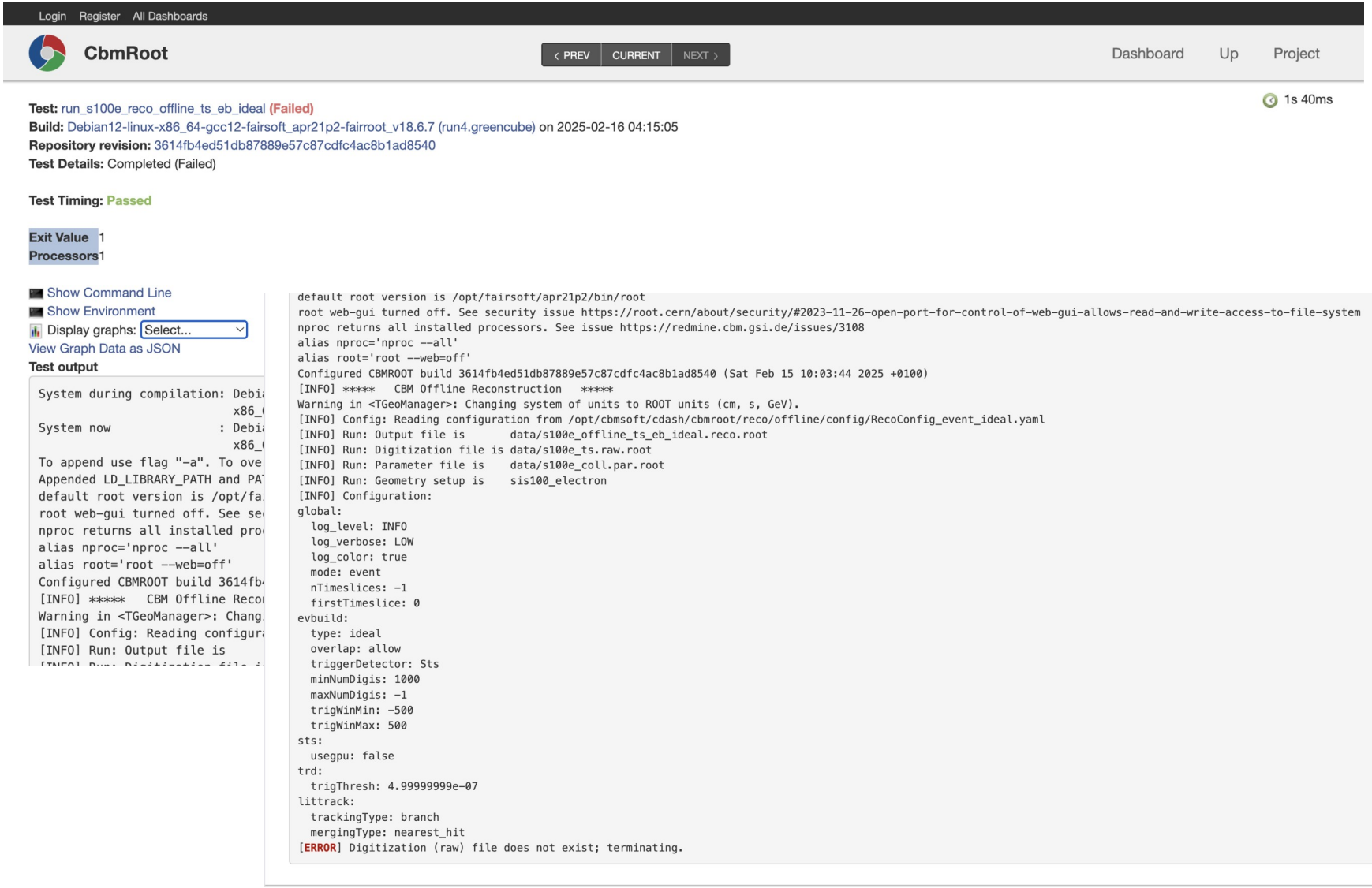
10 tests failed.

Name	Status	Time Status	Time	Details	History	Summary
c2f_reco_event_sis100_electron	Failed	Passed	13m 20s 30ms	Completed (Timeout)	Unstable	Unstable
mcbm_reco_l1tr_2022_2488	Failed	Passed	1h 59m 59s 620ms	Completed (Timeout)	Unstable	Broken
mcbm_reco_l1tr_digievent_2022_2488	Failed	Passed	1h 59m 59s 640ms	Completed (Timeout)	Unstable	Broken
OnlineRecoCanProcessTimeslice	Failed	Passed	26m 40s 160ms	Completed (Timeout)	Broken	Broken
PWG_common_production_run_json	Failed	Passed	11m 40s 50ms	Completed (Timeout)	Unstable	Unstable
run_s100c_reco_offline_ev_real	Failed	Passed	2m 44s 460ms	Completed (Failed)	Unstable	Unstable
run_s100e_reco_offline_ts_eb_ideal	Failed	Passed	1s 40ms	Completed (Failed)	Unstable	Unstable
run_s100e_reco_offline_ts_eb_real	Failed	Passed	1s 50ms	Completed (Failed)	Unstable	Unstable
run_s100e_reco_offline_ts_tb	Failed	Passed	1s 70ms	Completed (Failed)	Unstable	Unstable
run_s100e_tra_sign	Failed	Passed	6s 310ms	Completed (Failed)	Unstable	Unstable

CDash Failed Test Details



CDash Failed Test Details



Summary

- A little bit of CbmRoot history
  - More than 20 years of development
- Overview about complex infrastructure behind CbmRoot
  - Hope I could give you an idea about our compute infrastructure
  - Hope I don’t leave you more puzzled than before
- Many tools are used and needed to manage a large software project
  - Introduce some of them
- Tried to use as many available tools as possible
  - Keep own code for infrastructure minimal
  - Add glue code to connect existing tools

Outlook

- Your feedback is very welcome and actually needed
  - What can we do from the software project to help you doing your software related work?
  - Where in software and software infrastructure do you need more insight?
  - Do you think it is useful to have tutorials or hands on?
    - Which topics?
    - When and how often?