FairRoot
Build and Test System

Mohammad Al-Turany (IT-GSI)
Denis Bertini (IT-GSI)
Florian Uhlig (IT/CBM-GSI)

Outline

- What did/do we use.
- CMake
- Examples
- CTest/Dashboard

History/Motivation

- Start with self written Makefiles
  - Need work when porting to another platform
- Autotools (autoconf, automake, etc.)
  - Standard for *ix systems
  - Easy to use for user (./configure && make && make install)
  - Different macro languages for different tools in chain
  - „Autohell“ if there is a problem, even a blank character at the wrong position
  - No test system
- Cmake/Ctest/Dashboard

Cmake - What is it?

- Open sorce project (BSD style license)
- Family of tools to build, test and package software
- Meta build tool generates input for native tools
  - UNIX Makefile
  - Xcode
  - Visual Studio 6,7,8,9 IDE files
  - KDevelop
  - Eclipse
- Who is using it?
  - KDE, Scribus, SecondLife, ITK,VTK, FairRoot ;-)
- Who is behind Cmake
  - Kitware, Los Alamos National Labs, Sandia National Labs, National Library of Medcine, NAMIC

CMake Features

- Support complex custom commands
  - Generate code during build process which is then compiled (e.g. rootcint)
  - RuleChecker
  - Doxygen
- Optional component support (turn on/off features)
- Shared library and DLL support (version support)
- Single and simple input format for all platforms
- Automatic dependency generation (C, C++, Fortran)
  - Full dependencies: build a target in one directory, and everything this target depends on will be up to date
- Parallel builds (if supported by the native tool e.g. gmake -j4)
- Out of Source builds
- Linux, Mac OS X, SunOS, HPUX, IRIX, Windows, etc.
- Simple marco language
- Only depends on compiler and native build tool

Cmake Features (cont.)

- Color and progress output for make
- Automatic rerun of cmake if any cmake input file change
- Graphviz output for visualization of dependency trees
- Works with parallel make and on build farms
- make help shows all possible targets in the directory
- make foo.o build only foo.o and everything foo.o depends on
- CMake has a GUI layer for easy editing of input variables
- CMake has a command line interface
- Cross compiling support (CMake 2.6)

Hello world example for physicists

```
#CMakeLists.txt for simple test program
project(Tutorial)
add_executable(Tutorial tutorial.cxx)
```

```
/ A simple program that computes the square root of a number
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main (int argc, char *argv[])
{
  if (argc < 2)
    {
      fprintf(stdout,"Usage: %s number\n",argv[0]);
      return 1;
    }
  double inputValue = atof(argv[1]);
  double outputValue = sqrt(inputValue);
  fprintf(stdout,"The square root of %g is %g\n",
          inputValue, outputValue);
  return 0;
}
```

If live demo fails

uhlig@lxi006: /misc/uhlig — bash — 80×24

bash | bash

```
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$
localhost:build uhlig$ cmake ..
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Check size of void*
-- Check size of void* - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/uhlig/Workshop/demo1/build
localhost:build uhlig$
```

uhlig@lxi006: /misc/uhlig — ccmake — 80×24

ccmake | bash

```
                                                    Page 1 of 1
CMAKE_BACKWARDS_COMPATIBILITY    2.4
CMAKE_BUILD_TYPE
CMAKE_INSTALL_PREFIX             /usr/local
CMAKE_OSX_ARCHITECTURES          i386
CMAKE_OSX_SYSROOT                /Developer/SDKs/MacOSX10.4u.sdk
EXECUTABLE_OUTPUT_PATH
LIBRARY_OUTPUT_PATH

CMAKE_BACKWARDS_COMPATIBILITY: For backwards compatibility, what version of CMak
Press [enter] to edit option                    CMake Version 2.4 - patch 7
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Main config file for CbmRoot

```
# Check if cmake has the required version
CMAKE_MINIMUM_REQUIRED(VERSION 2.4.3 FATAL_ERROR)

# Set name of our project to "CBMROOT". Has to be done
# after check of cmake version
project(CBMROOT)

# where to look first for cmake modules, before ${CMAKE_ROOT}/Modules/
# is checked
set(CMAKE_MODULE_PATH "${CMAKE_SOURCE_DIR}/cmake/modules")

# Load some basic macros which are needed later on
include(CbmMacros)
include(WriteConfigFile)
include(Dart)
include(CheckCompiler)

#Check the compiler and set the compile and link flags
Check_Compiler()

# Check if the user wants to build the project in the source
# directory
CHECK_OUT_OF_SOURCE_BUILD()

# Check if we are on an UNIX system. If not stop with an error
# message
IF(NOT UNIX)
  MESSAGE(FATAL_ERROR "You're not on an UNIX system. The project was up
    to now only tested on UNIX systems, so we break here. If you want to go on
   please edit the CMakeLists.txt in the source directory.")
ENDIF(NOT UNIX)

IF(NOT DEFINED ENV{SIMPATH})
      MESSAGE(FATAL_ERROR "You did not define the environment
      variable SIMPATH which is nedded to find the external packages.
     Please set this variable and execute cmake again.")
ENDIF(NOT DEFINED ENV{SIMPATH})
CHECK_EXTERNAL_PACKAGES_INSTALLATION()
```

```
find_package(ROOT REQUIRED)
find_package(PLUTO REQUIRED)
find_package(GENERATORS REQUIRED)
find_package(GEANT3 REQUIRED)
find_package(GEANT4)
find_package(GEANT4VMC)
find_package(CLHEP)
find_package(RuleChecker)

# Set the library version in the main CMakeLists.txt
SET(CBMROOT_MAJOR_VERSION 0)
SET(CBMROOT_MINOR_VERSION 0)
SET(CBMROOT_PATCH_VERSION 0)
SET(CBMROOT_VERSION
"${CBMROOT_MAJOR_VERSION}.${CBMROOT_MINOR_VERSION}.${CBMROOT_PATCH_VERSION}")
SET(CBMROOT_LIBRARY_PROPERTIES ${CBMROOT_LIBRARY_PROPERTIES}
    VERSION "${CBMROOT_VERSION}"
    SOVERSION "${CBMROOT_MAJOR_VERSION}"
)

# Recurse into the given subdirectories.  This does not actually
# cause another cmake executable to run.  The same process will walk through
# the project's entire directory structure.
add_subdirectory (base)
...
add_subdirectory(zdc)

if(GEANT4_FOUND AND GEANT4VMC_FOUND AND CLHEP_FOUND)
  add_subdirectory (cbmg4)
endif(GEANT4_FOUND AND GEANT4VMC_FOUND AND CLHEP_FOUND)

Option(BUILD_DOXYGEN "Build Doxygen" OFF)
if(BUILD_DOXYGEN)
  MESSAGE(STATUS "*** Building the Doxygen documentaion ***")
  ADD_SUBDIRECTORY(doxygen)
endif(BUILD_DOXYGEN)

If(RULE_CHECKER_FOUND)
  ADD_CUSTOM_TARGET(RULES
    COMMAND ${RULE_CHECKER_SCRIPT1} ${CMAKE_BINARY_DIR} viol > violations.html)
ENDIf(RULE_CHECKER_FOUND)

WRITE_CONFIG_FILE(config.sh)
WRITE_CONFIG_FILE(config.csh)
```

Template for subproject

```
# Create a library called "lib<name>" which includes the source files given in
# the array .
# The extension is already found.  Any number of sources could be listed here.

set(INCLUDE_DIRECTORIES
${ROOT_INCLUDE_DIR}
${CBMROOT_SOURCE_DIR}/geobase
...
)

include_directories( ${INCLUDE_DIRECTORIES})

set(LINK_DIRECTORIES
${ROOT_LIBRARY_DIR}
)

link_directories( ${LINK_DIRECTORIES})

set(<name>_SRCS
Source_File_1.cxx
....
)

If(RULE_CHECKER_FOUND)
  CHECK_RULES(„${<name>_SRCS}“ „${INCLUDE_DIRECTORIES}“ TRD_RULES)
endIf(RULE_CHECKER_FOUND)

# fill list of header files from list of source files
# by exchanging the file extension
CHANGE_FILE_EXTENSION(*.cxx *.h <name>_HEADERS "${<name>_SRCS}")

set(<name>_LINKDEF <name>LinkDef.h)
set(<name>_DICTIONARY ${CMAKE_CURRENT_BINARY_DIR}/<name>Dict.cxx)

ROOT_GENERATE_DICTIONARY("${<name>_HEADERS}" "${<nmae>_LINKDEF}" "${<name>_DICTIONARY}" "${INCLUDE_DIRECTORIES}")

set(<name>_SRCS ${<name>_SRCS} ${<name>_DICTIONARY})

add_library(<name> SHARED ${<name>_SRCS})
target_link_libraries(<name> ${ROOT_LIBRARIES})
set_target_properties(<name> PROPERTIES  ${CBMROOT_LIBRARY_PROPERTIES})

################ install ##################
install(TARGETS <name> DESTINATION ${CMAKE_BINARY_DIR}/lib)
```

Macros

```
MACRO (ROOT_GENERATE_DICTIONARY INFILES LINKDEF_FILE OUTFILE INCLUDE_DIRS_IN)

  set(INCLUDE_DIRS)

  foreach (_current_FILE ${INCLUDE_DIRS_IN})
    set(INCLUDE_DIRS ${INCLUDE_DIRS} -I${_current_FILE})
  endforeach (_current_FILE ${INCLUDE_DIRS_IN})

STRING(REGEX REPLACE "^(.*)\\.(.*)$" "\\1.h" tmp "${OUTFILE}")
  SET (OUTFILES ${OUTFILE} ${tmp})

  ADD_CUSTOM_COMMAND(OUTPUT ${OUTFILES}
     COMMAND ${ROOT_CINT_EXECUTABLE}
     ARGS -f ${OUTFILE} -c -DHAVE_CONFIG_H ${INCLUDE_DIRS} ${INFILES} ${LINKDEF_FILE} DEPENDS ${INFILES})

ENDMACRO (ROOT_GENERATE_DICTIONARY)
```

Why test?

- If it is not tested, it does not work !
- With good testing global changes are much easier and saver
- Large code base ist to large/complicated for a single developer to understand and maintain
- Identify problems when they occor
- FairRoot depends on external packages which can cause problems
- Direct feedback for the developers as they experiment with new features

How to test?

- Use CTest
- Tests are easy to set up
  - Add enable_testing() in main CMakeLists.txt
  - add_test(name executable arg1 arg2 ...)
- Run ***make test*** in build directory
- Use ctest (shipped with cmake) to run tests
- Run tests on all supported platforms
- Show results of tests on dashboard
- We run root marcos as test
  - add_test(run_sim ${ROOTSYS}/bin/root -b -l ${CBMROOT_SOURCE_DIR}/macro/run/run_sim.C)

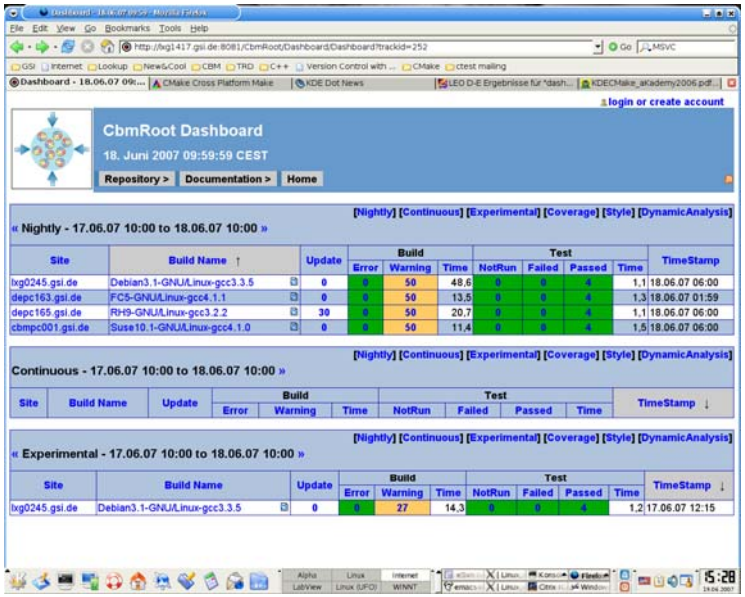
Automatic tests

- Update source from SVN repository
- Configure project
- Generate Makefiles
- Build the project
- Run the tests
- Submit results to a webserver (Dashboard)
  - Run ***make Experimental/Nightly*** in build directory
- Run the chain automatically if there is a commit to the repository

Automatic tests (cont.)

SVN maintains source code revision

CTest/CMake compiles and test the newly commited source code on distributed clients



Typical developer checks in code

Developer reviews the results

Dashboard

- Client/Server architecture
  - Test on all platforms available to users
  - User use scripts to run the test and submit only results
- Different tests
  - Experimental
    - No update from repository
    - Usefull to test code before commit
  - Nightly
    - Update from repository with given timestamp
    - All machines should run with the same code base
  - Continuous
    - Should run whenever there is a change in the repository
  - Coverage/MemCheck
    - Check the code coverage of the tests
    - Runs valgrind to check for memeory leaks

Dashboard

Dashboard (how it looks like)

Dashboard – 18.01.08 08:59

http://lxg1417.gsi.de:8081/CbmRoot/Dashboard/Dashboard?trackid=1398

Apple Yahoo! Google Maps YouTube Wikipedia News (562) Beliebt

login or create account

CbmRoot Dashboard

18. Januar 2008 08:59:59 CET

Repository > | Documentation > | RuleChecker | Home

[Nightly] [Continuous] [Experimental] [Coverage] [Style] [DynamicAnalysis]

« Nightly - 17.01.08 09:00 to 18.01.08 09:00 »

Site	Build Name ↑	Update	Build			Test				TimeStamp
			Error	Warning	Time	NotRun	Failed	Passed	Time	
lxetch32.gsi.de	Debian/Etch-GNU/Linux-i686-gcc4.1.2	0	0	0	13,3	0	1	3	0,9	18.01.08 05:00
lxetch32.gsi.de	Debian/Etch-GNU/Linux-i686-gcc4.1.2-devel	4	0	0	12,6	0	1	3	1,2	18.01.08 05:00
lxetch64.gsi.de	Debian/Etch-GNU/Linux-x86_64-gcc4.1.2	0	0	2	12,5	0	1	3	1,4	18.01.08 05:00
lxetch64.gsi.de	Debian/Etch-GNU/Linux-x86_64-gcc4.1.2-devel	4	0	2	11,8	0	1	3	1,2	18.01.08 05:00
lxg1417.gsi.de	Debian/Sarge-GNU/Linux-i686-gcc3.3.5	8	0	0	25,0	0	1	3	2,5	18.01.08 05:00
lxg1417.gsi.de	Debian/Sarge-GNU/Linux-i686-gcc3.3.5-devel	0	0	0	25,1	0	1	3	3,0	18.01.08 05:00
lxsarge64.gsi.de	Debian/Sarge-GNU/Linux-x86_64-gcc3.3.5	0	0	0	13,9	0	1	3	1,1	18.01.08 05:00
lxsarge64.gsi.de	Debian/Sarge-GNU/Linux-x86_64-gcc3.3.5-devel	0	0	0	14,6	0	1	3	1,6	18.01.08 05:00
fwklux2.fz-rossendorf.de	Debian-GNU/Linux-gcc4.1.3	2	0	1	15,1	0	1	3	1,6	18.01.08 05:00
depc163.gsi.de	FC5-GNU/Linux-gcc4.1.1	0	0	1	20,7	0	1	3	2,2	18.01.08 00:59
depc163.gsi.de	FC5-GNU/Linux-icpc10.0	2	0	5	50,7	0	3	1	1,0	18.01.08 00:59
cbmpc001.gsi.de	Suse10.1-GNU/Linux-i686-gcc4.1.0	2	0	1	17,7	0	1	3	1,5	18.01.08 05:00

[Nightly] [Continuous] [Experimental] [Coverage] [Style] [DynamicAnalysis]

« Continuous - 17.01.08 09:00 to 18.01.08 09:00 »

Site	Build Name	Update	Build			Test				TimeStamp ↓
			Error	Warning	Time	NotRun	Failed	Passed	Time	

[Nightly] [Continuous] [Experimental] [Coverage] [Style] [DynamicAnalysis]

« Nightly - 17.06.07 10:00 to 18.06.07 10:00 »

Site	Build Name ↑	Update	Build			Test				TimeStamp
			Error	Warning	Time	NotRun	Failed	Passed	Time	
lxg0245.gsi.de	Debian3.1-GNU/Linux-gcc3.3.5	0	0	50	48,6	0	0	4	1,1	18.06.07 06:00
depc163.gsi.de	FC5-GNU/Linux-gcc4.1.1	0	0	50	13,5	0	0	4	1,3	18.06.07 01:59
depc165.gsi.de	RH9-GNU/Linux-gcc3.2.2	30	0	50	20,7	0	0	4	1,1	18.06.07 06:00
cbmpc001.gsi.de	Suse10.1-GNU/Linux-gcc4.1.0	0	0	50	11,4	0	0	4	1,5	18.06.07 06:00

ElapsedTime	0.1
EndDateTime	Jun
StartDateTime	Jun
UpdateCommand	"/usr
UpdateType	SVN

	_Properties	Passed	
	run	Passed	21,212
	run_sim	Passed	34,798
	run_reco	Passed	13,092

Chart times

Changed files as of

[Expand all | Collapse all]

Updated files (30)

rich/CbmRichMatchRings.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

rich/CbmRichRingFinderTrack.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

rich/CbmRichRingTrackAssignIdeal.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

rich/CbmRichLightSpot.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

rich/CbmRichAnalysisHits.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

rich/CbmRichRingFitterTAU.cxx Revision: 918 by uhlig
Correct small problems resulting in compiler warnings

Build Warnings (50)

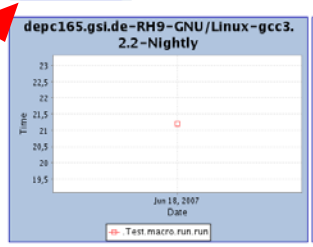
Warning Build Log Line 145

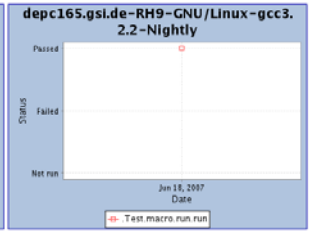
File: rich/CbmRichLightSpot.cxx Line: 515

```
[ 22%] Built target Passive
[ 22%] Generating CbmRichDict.cxx, CbmRichDict.h
Scanning dependencies of target Rich
[ 22%] Building CXX object rich/CMakeFiles/Rich.dir/CbmGeoRich.o
[ 22%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichLightSpot.o
/.../cbmroot/rich/CbmRichLightSpot.cxx: In member function 'void CbmRichLightSpot::RingRobus
/.../cbmroot/rich/CbmRichLightSpot.cxx:515: warning: statement has no effect
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichRing.o
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichAnalysisHits.o
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichRingFinder.o
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRich.o
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichRingFinderImp.o
[ 23%] Building CXX object rich/CMakeFiles/Rich.dir/CbmRichHit.o
```

Site Name: depc165.gsi.de
Build Name: RH9-GNU/Linux-gcc3.2.2
Test Name: .Test.macro.run.run Passed

Command Line	/home/florian/cbmsoft1/cbmsoft/tools/root/bin/root -b -l /home/florian/SVN/ctest/cbmroot/macro/run/run.C
Completion Status	Completed
Execution Time	21.2121





Test output

```
Processing /home/florian/SVN/ctest/cbmroot/macro/run/run.C...
Note: File "/home/florian/cbmsoft1/cbmsoft/tools/root/lib/libPhysics.so" already loaded
Note: File "/home/florian/cbmsoft1/cbmsoft/tools/root/lib/libEG.so" already loaded
Error in <TUnixSystem::DynamicPathName>: libCbmBase[.so | .sl | .dl | .a | .dll] does not e

PSaid instance created...  access via gSaid->f()

- RTDB container factory CbmBaseContFact
- RTDB container factory CbmFieldContFact
- RTDB container factory CbmPassiveContFact
- RTDB container factory CbmStsContFact
```

Rule Checker

http://lxg1417.gsi.de:8081/PandaRoot/violations/101207/violations.html

PandaRoot Coding Convention Violation Table

Rules	GC1	GC2	RC3	RC4	RC5	RC6	RC7	RC8	RC9	RC10	RC11	RC12	RC14	RC15	RC16	RN3	RN4	RN5	RN6	RN9	RN11	RN13	RN15	RN17	RN19	RN22	RS1	RS2	RS3	RS4	RS5	TOTAL
base	14	31	1	0	0	4	3	0	0	13	55	18	4	0	2	28	0	0	1	41	25	29	0	2	7	0	38	104	111	3	5	539
geobase	0	20	1	1	1	20	0	0	0	15	57	206	1	0	0	2	1	1	0	330	81	1	0	21	2	0	43	18	22	0	0	844
generators	3	13	0	0	0	0	1	0	0	6	1	0	0	0	0	1	0	0	0	0	1	2	0	0	0	0	8	1	20	0	0	57
parbase	1	16	5	4	4	14	1	0	0	12	15	164	1	0	0	0	4	4	0	209	40	0	1	17	1	0	21	2	33	0	0	569
mcstack	5	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	10	16	0	0	38
field	2	15	3	0	0	0	2	0	2	5	12	0	0	0	0	0	0	0	0	10	0	8	0	0	0	0	27	56	68	0	0	210
pgenerators	0	5	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0	9	0	0	23
stt	30	40	5	0	0	4	2	0	5	11	68	14	4	0	3	83	0	0	0	25	121	169	10	1	0	0	34	125	95	0	0	849
tpc	149	113	72	3	1	43	1	24	17	40	32	48	2	5	0	528	10	8	16	282	444	37	0	5	3	0	79	315	157	2	2	2438
mvd	65	50	5	0	0	0	1	0	0	16	18	0	0	0	0	1	0	0	0	20	28	22	0	0	0	0	50	137	157	0	0	570
muo	0	11	1	0	0	2	1	0	1	3	13	5	11	0	11	6	0	0	0	19	26	0	0	0	0	0	14	23	20	0	0	167
emc	73	63	21	4	0	5	3	3	2	17	59	37	24	0	5	140	4	4	1	94	85	19	0	45	12	0	29	204	92	0	0	1045
drc	0	9	0	0	0	2	1	0	0	2	17	5	2	0	12	0	0	0	0	7	8	0	0	0	0	0	6	36	9	0	0	116
hyp	0	12	0	0	0	3	1	0	1	3	11	3	5	0	4	0	0	0	0	6	8	8	0	0	0	0	7	19	20	0	0	111
genfit	252	30	17	0	1	11	0	2	0	5	24	27	0	0	0	65	1	1	18	172	51	27	0	0	2	0	24	45	43	21	49	888
recotasks	46	20	12	0	0	11	0	0	3	9	2	0	0	0	0	90	0	0	11	25	89	2	0	0	1	0	12	82	19	0	0	434
dch	15	22	4	0	0	4	1	0	4	8	6	7	2	0	0	0	0	0	0	7	2	10	1	5	4	0	20	39	46	0	0	207
tof	0	11	0	0	0	2	1	0	1	3	13	3	8	0	4	0	0	0	0	6	8	4	0	0	0	0	7	24	20	0	0	115
lhetrack	0	17	9	0	0	10	0	0	0	7	26	0	0	0	3	87	0	0	1	0	1	38	0	3	2	0	16	28	17	0	0	265
trackbase	9	6	3	0	0	0	0	0	0	0	47	0	0	0	0	25	0	0	0	0	4	328	0	0	0	0	6	18	22	0	0	468
trackrep	17	2	1	0	0	1	0	0	0	1	2	2	0	0	0	1	0	0	1	12	1	0	0	0	0	0	1	1	2	0	0	45
tpc/tpcreco	169	26	15	0	0	7	0	0	6	8	26	0	0	0	0	103	0	0	1	112	87	14	0	0	1	0	15	74	58	0	0	722
Total	**850**	**536**	**175**	**12**	**7**	**143**	**19**	**29**	**42**	**188**	**504**	**539**	**64**	**5**	**44**	**1160**	**20**	**18**	**50**	**1377**	**1113**	**718**	**12**	**99**	**35**	**0**	**462**	**1361**	**1056**	**26**	**56**	**10720**

Summary and Outlook

- CMake/Ctest/Dashboard has all features we need
- Easy to extend to our needs (e.g. RuleChecker)
- Everything is Open Source and can be changed if needed

- Use SVN to trigger Continuous build of project
- Use CDash (successor of Dashboard)
  - Php scripts which run on a webserver
  - Mysql database
  - More functionality
    - Send email to developer who breaks the dashboard
    - Better administrative tools
- Implement Coverage tests/Memory checks
- Try to incorporate the RuleChecker in CDash