ROOT: High Quality, Systematically

Axel Naumann <axel@cern.ch> (CERN PH-SFT)

Introduction

ROOT's 2.5 million lines of code saw about 5000 source changes (revisions) in 2011. ROOT is a fundamental software ingredient for about 20000 users for their data analysis, and for almost all HEP experiments e.g. for their data. One of ROOT's most important properties is thus its reliability and quality, on the wide range of supported platforms.

To ensure code quality, ROOT has employed several tools over the past year that monitor aspects from source code quality to test results. It has pioneered new approaches at CERN that have already or are currently spreading outside of the ROOT project. This poster shows the scrutiny a source code change goes through and the new features of our build tool, that are available also for users of ROOT.

Build Automation: Electric Commander

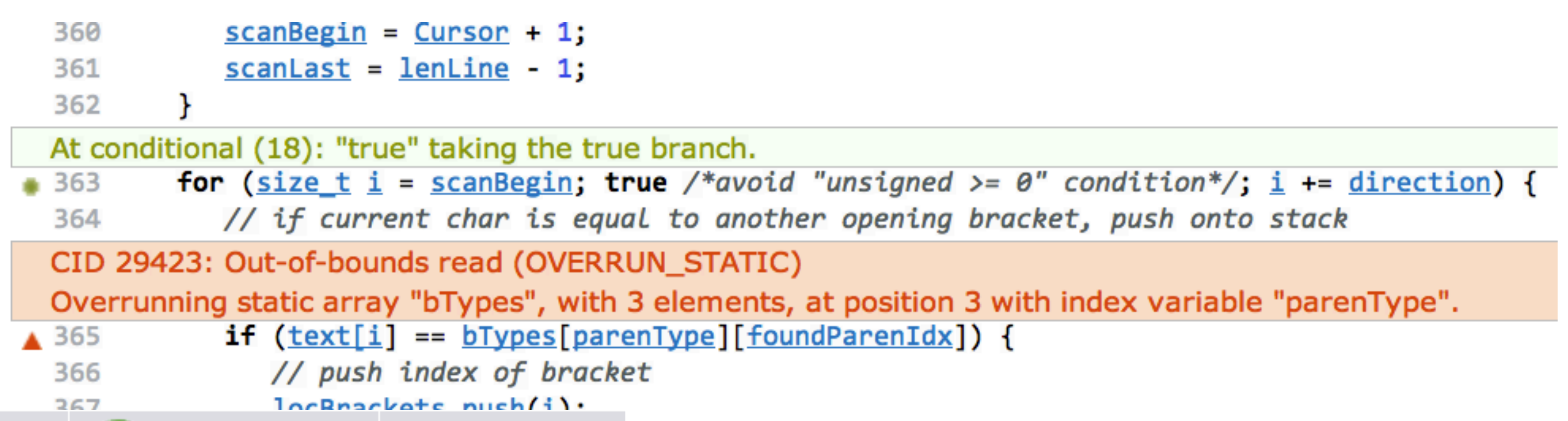
ROOT uses "Electric Commander" (EC), an extremely reliable, web-based, multi-platform build automation tool that starts build jobs, parses output, and then stores build results in a database. We use it to build ROOT on 14 platforms:

- ☑ public release snapshots every night, EC builds binary releases of ROOT, test them, and publishes them at https://ecsft.cern.ch/dist
- ☑ continuous integration EC triggers an incremental build on all platforms, developers get notified of build warnings and errors.
- ☑ hourly builds EC runs complete builds and tests of ROOT every two hours

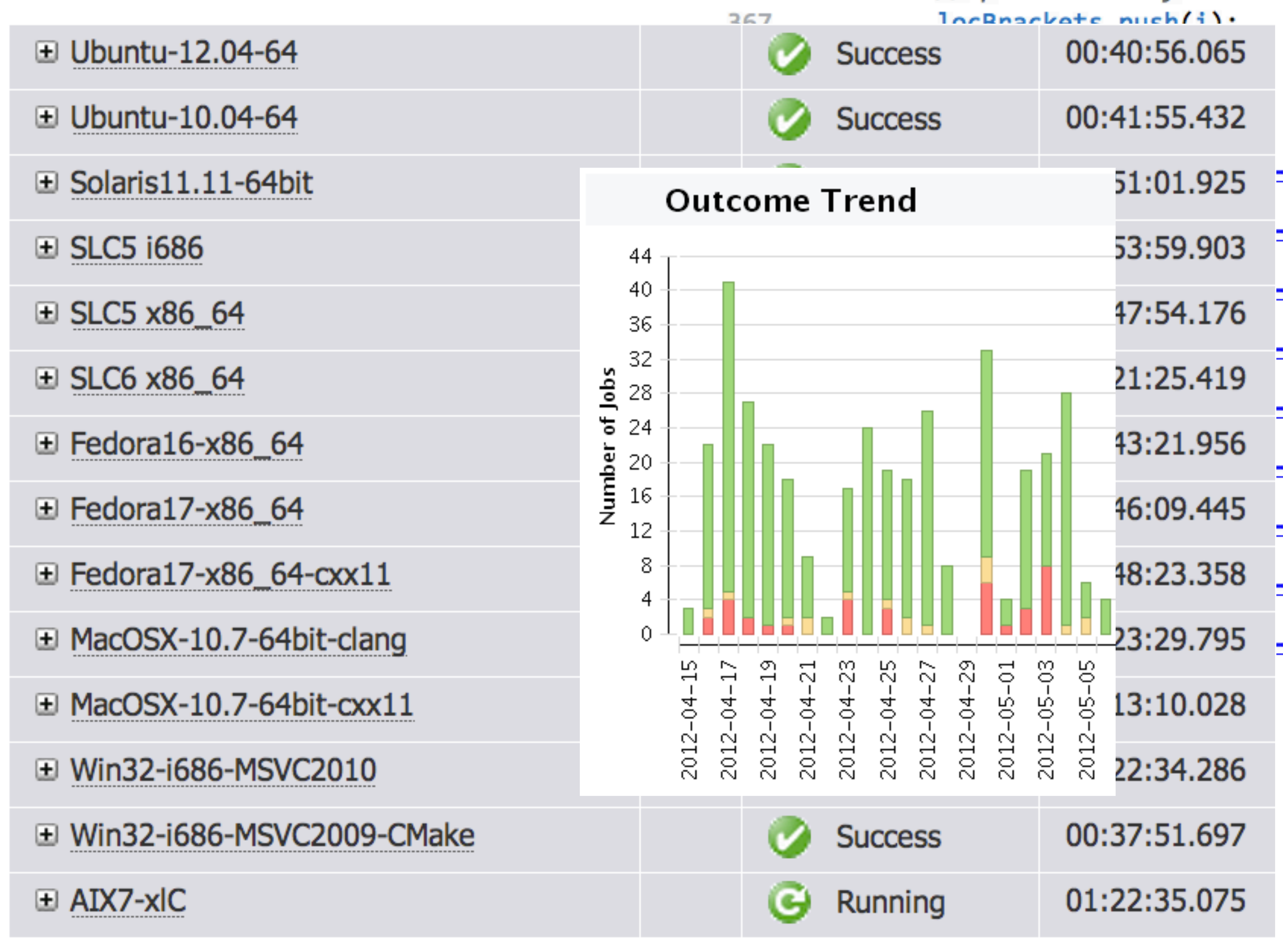
Code Checkers

ROOT's code gets checked on a daily basis in several ways:

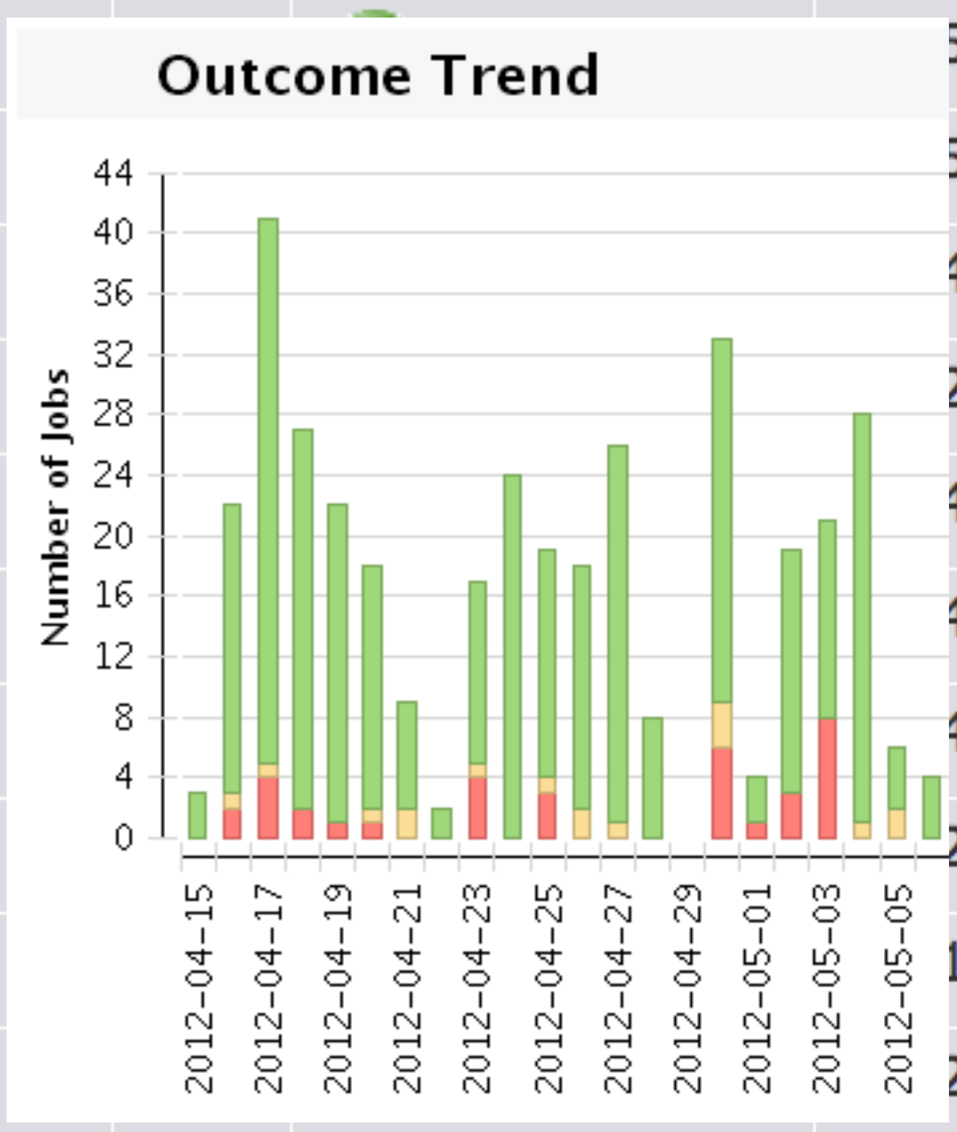
- ☑ static analysis Coverity finds coding flaws (possible use after delete etc) by analyzing the source code, not the runtime behavior
- ☑ ROOT's coding rules checked with eclair, a tool based on the open-source compiler clang / LLVM
- ☑ multi-platform last but not least, building ROOT on 14 platforms with 13 different compilers / versions gives warnings for problematic code

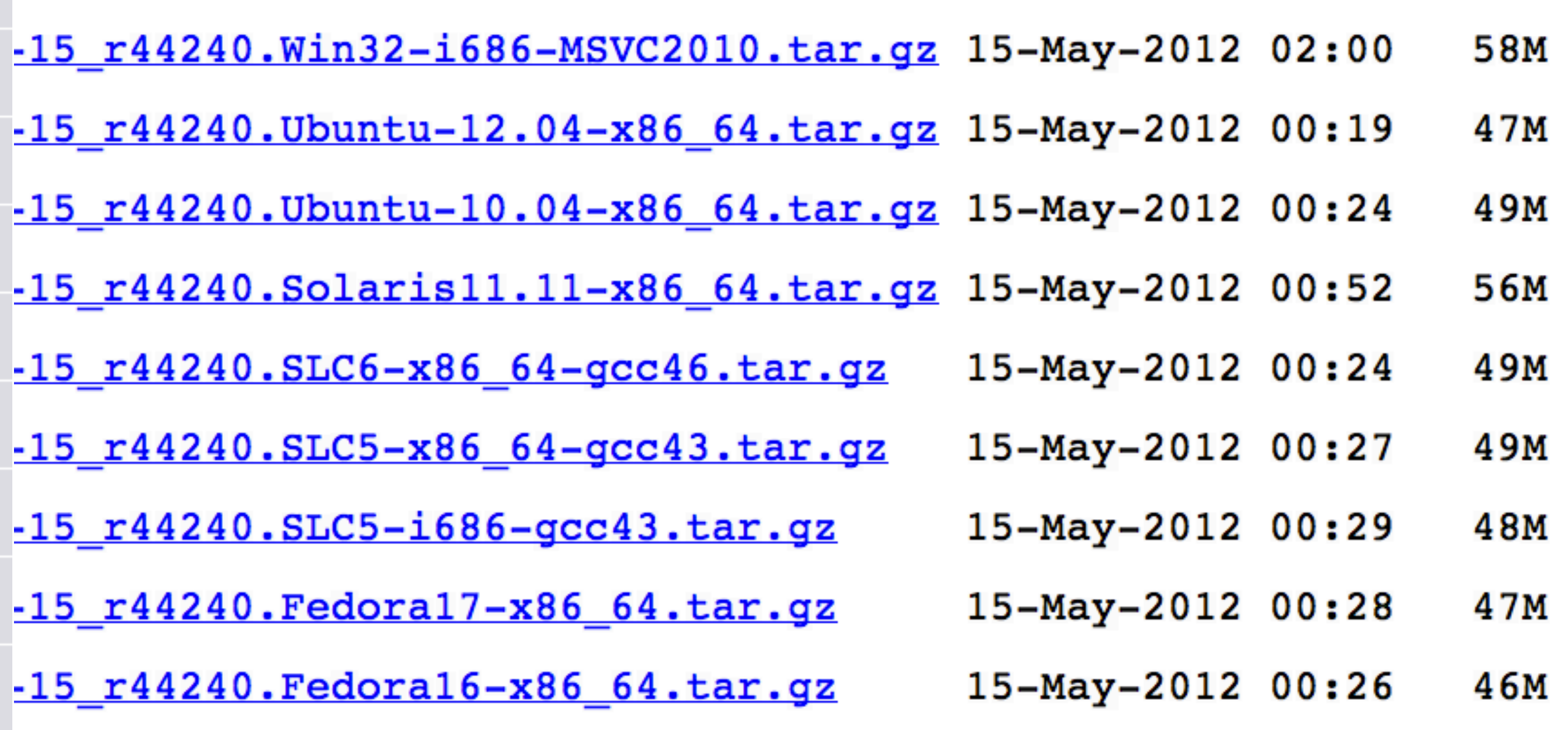


```
        scanBegin = Cursor + 1;
        scanLast = lenLine - 1;
    }
At conditional (18): "true" taking the true branch.
    for (size_t i = scanBegin; true /*avoid "unsigned >= 0" condition*/; i += direction) {
       // if current char is equal to another opening bracket, push onto stack
CID 29423: Out-of-bounds read (OVERRUN_STATIC)
Overrunning static array "bTypes", with 3 elements, at position 3 with index variable "parenType".
       if (text[i] == bTypes[parenType][foundParenIdx]) {
          // push index of bracket
          locBrackets.push(i);
```

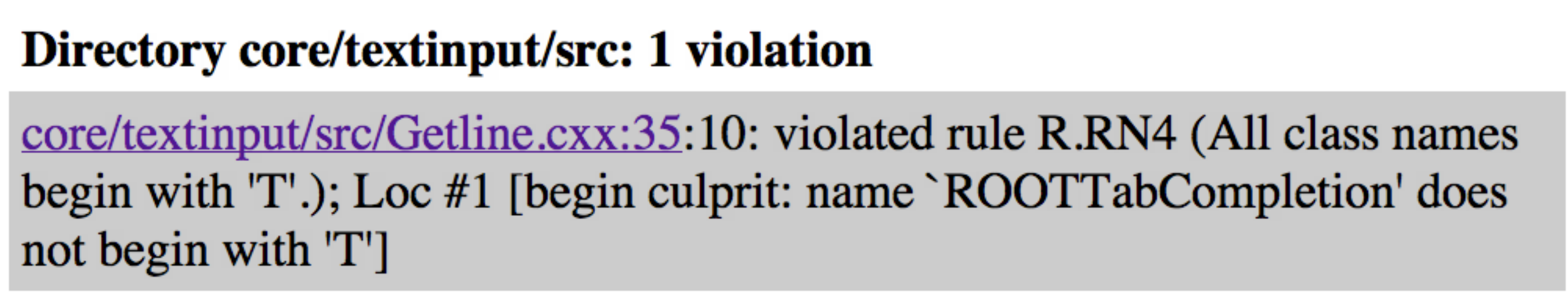


⊞ Ubuntu-12.04-64	✓ Success	00:40:56.065
⊞ Ubuntu-10.04-64	✓ Success	00:41:55.432
⊞ Solaris11.11-64bit	[illegible]	51:01.925
⊞ SLC5 i686	[illegible]	53:59.903
⊞ SLC5 x86_64	[illegible]	47:54.176
⊞ SLC6 x86_64	[illegible]	21:25.419
⊞ Fedora16-x86_64	[illegible]	43:21.956
⊞ Fedora17-x86_64	[illegible]	46:09.445
⊞ Fedora17-x86_64-cxx11	[illegible]	48:23.358
⊞ MacOSX-10.7-64bit-clang	[illegible]	23:29.795
⊞ MacOSX-10.7-64bit-cxx11	[illegible]	13:10.028
⊞ Win32-i686-MSVC2010	[illegible]	22:34.286
⊞ Win32-i686-MSVC2009-CMake	✓ Success	00:37:51.697
⊞ AIX7-xlC	Running	01:22:35.075





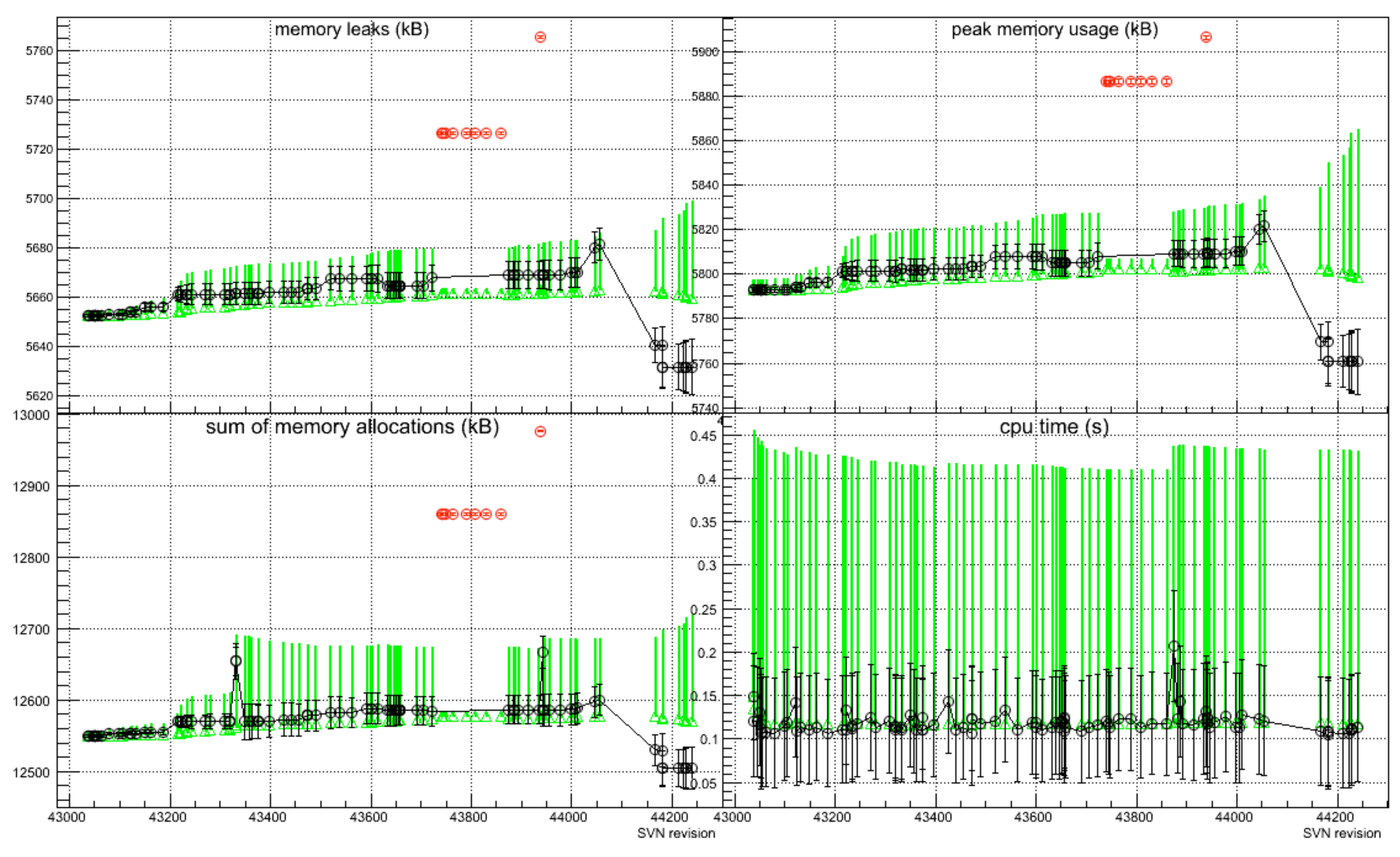
```
-15_r44240.Win32-i686-MSVC2010.tar.gz 15-May-2012 02:00   58M
-15_r44240.Ubuntu-12.04-x86_64.tar.gz 15-May-2012 00:19   47M
-15_r44240.Ubuntu-10.04-x86_64.tar.gz 15-May-2012 00:24   49M
-15_r44240.Solaris11.11-x86_64.tar.gz 15-May-2012 00:52   56M
-15_r44240.SLC6-x86_64-gcc46.tar.gz   15-May-2012 00:24   49M
-15_r44240.SLC5-x86_64-gcc43.tar.gz   15-May-2012 00:27   49M
-15_r44240.SLC5-i686-gcc43.tar.gz     15-May-2012 00:29   48M
-15_r44240.Fedora17-x86_64.tar.gz     15-May-2012 00:28   47M
-15_r44240.Fedora16-x86_64.tar.gz     15-May-2012 00:26   46M
```



Directory core/textinput/src: 1 violation

core/textinput/src/Getline.cxx:35:10: violated rule R.RN4 (All class names begin with 'T'.); Loc #1 [begin culprit: name `ROOTTabCompletion' does not begin with 'T']

root/hist/formula/runconstargs.C+



Performance Monitoring

We have implemented a custom (but reusable) tool for monitoring the performance of ROOT. It is profiling ROOT's test suite roottest, and triggers a test error if the deterioration in any measure is statistically significant. ROOT's performance is now monitored in these dimensions:

- ☑ CPU time, to detect new code that causes inefficiencies
- ☑ memory leaks, to detect missing deallocations
- ☑ maximum amount of memory allocated at any time during the process lifetime, to detect an increase in memory requirements
- ☑ sum of memory allocations over the whole process lifetime, to detect excessive allocations (even if the allocation gets freed later) that could cause excess memory churn contributing to memory fragmentation

Our implementation runs the binary to be monitored as a child process, pre-loading a library that captures all heap operations by symbol injection. That library sends the results of the monitoring to the parent process when the process terminates. The parent process then compares the results with earlier measurements. If the deviation is above a defined number of standard deviations, the parent process will report an error.

Summary

ROOT's trunk is not an adventure: it is promised to work. ROOT can now ensure that, by automatizing both build and quality checks. This makes developers and users more courageous: developers can dare to check-in fundamental changes because they will know immediately if something broke. And users can try the newest ROOT when they want, not when the experiment's framework is ready.
The success story of these QA building blocks for ROOT caused and causes them to spread across CERN: Coverity is now also used by e.g. Geant4, many CERN experiments and parts of the beam software for code analysis; Electric Commander is already employed for CernVM's cvmfs and soon for Geant4 and the experiments' common libraries (SPI).

Resources and further reading

1. ROOT's nightly release snapshots: https://ecsft.cern.ch/dist
2. ROOT's performance tracking results: http://www-root.fnal.gov/roottest/root-today/rootsrv1-trunk.data/
3. ROOT's build status (login "user", password "pass"): https://ecsft.cern.ch
4. Electric Commander: http://electriccloud.com/products/electriccommander.php
5. Coverity Static Analysis: http://coverity.com/products/static-analysis.html
6. Eclair, ROOT's coding rule checker: http://bugseng.com/products/eclair

More info: