root-project / **rootbench**

Collection of benchmarks and performance monitoring applications

101 commits | **1** branch | **0** releases | **9** contributors | LGPL-2.1

Branch: master | New pull request | Find file | Clone or download

oshadura Updating code to provide rb_unreachable functionality in debug/not de... … Latest commit 4326cad 11 days ago

cmake/modules	Fixing insource-ROOT compilation of rootbench	12 days ago
include/rootbench	Updating code to provide rb_unreachable functionality in debug/not de...	11 days ago
lib	Add libRBSupport.	13 days ago
root	Fix gcc builds which has no __has_builtin macro.	13 days ago
.clang-format	Add ROOT's clang-format and clang-tidy config.	5 months ago
.clang-tidy	Add ROOT's clang-format and clang-tidy config.	5 months ago
.gitignore	Update .gitignore	7 months ago
.travis.yml	Updating rootexternals for latest version	21 days ago
CMakeLists.txt	Fixing insource-ROOT compilation of rootbench	12 days ago
CTestConfig.cmake	Integration rootbench (#1)	6 months ago
CTestCustom.cmake	Integration rootbench (#1)	6 months ago
LICENSE	Initial commit	7 months ago
README.md	Update README.md	23 days ago

README.md

ROOT Benchmarks

This repository contains a set of relatively small programs (usually based on gbenchmark micro benchmarking infrastructure) built on top of ROOT. Their primary goal is to provide stable performance metrics which can be monitored over time.

Project Health

Linux/OSX build passing	Experimental Benchmark Coverage: coverage 3%

About

Collection of benchmarks and performance monitoring applications

Building

ROOTBench can be built standalone and as part of ROOT. If you want to enable ROOTBench for ROOT just add the `-Drootbench=On` option to your cmake configuration.

Building ROOTBench standalone

ROOTBench should be able to find ROOT at configuration time. Make sure you ran `source $ROOTSYS/bin/thisroot.sh` .

```
git clone https://github.com/root-project/rootbench.git
mkdir build
cd build
cmake ../rootbench
cmake --build . -- -j4
```

Extending the benchmarks

ROOTBench relies on Google Benchmark. We recommend to read the available documentation and browse the existing examples here for more advanced usage.

Background

This repository is being integrated in two steps:

- We run TravisCI on each pull request -- the public infrastructure is time limited and we use the latest ROOT nightly build available in CVMFS and EOS. This way we can integrate with public services such as Coveralls. Based on the TravisCI information we compute the benchmarking coverage of ROOTBench against ROOT. The idea is to make sure that we have well-distributed benchmarking coverage.
- We run on dedicated CERN **OpenLab** machines twice a day -- we build ROOT and ROOTBench from scratch and collect performance data. The data is uploaded to our Grafana service available here (requires CERN login).

The integration process depends on the overall benchmarking time. Contributors are encouraged to write well-focused microbenchmarks ensuring good benchmarking coverage. Non-overlapping microbenchmarks seem to be the only reasonable way to control the pressure on the infrastructure.

Conventions

There are several practical conventions that we should follow:

- Coding conventions -- ROOTBench follows the coding conventions of ROOT to a great extent.
- The routines used for benchmarking shall have the following names `BM_CLASSNAME_ROUTINE` -- the `BM` prefix allows us (or tools) to easily identify which is the main benchmarking function.

Simple benchmark template

Add file called CLASSNAMEBenchmarks.cxx where CLASSNAME is the name of the ROOT class we benchmark.

```
#include "ROOT_HEADER_TO_BENCHMARK.h"

#include "benchmark/benchmark.h"

// Replace the CLASSNAME and ROUTINE with the ROOT class and routine you are benchmarking respectively.
static void BM_CLASSNAME_ROUTINE(benchmark::State &state) {
  // Initialization section before actual benchmarking.
  for (auto _ : state) {
    // The benchmarking code goes here.
  }
  // Teardown.
}
BENCHMARK(BM_CLASSNAME_ROUTINE);

// In the end of the file we add our main().
BENCHMARK_MAIN();
```

Register the benchmark in the system. Add an entry to the `CMakeLists.txt` next to the source code of the benchmark.

```
RB_ADD_GBENCHMARK(CLASSNAMEBenchmarks
  CLASSNAMEBenchmarks.cxx
  LABEL short
  LIBRARIES LIST OF LIB DEPENDENCIES)
```

This is a very basic working example. If you need extra functionality please read the Google Benchmark Docs.