Enhancements to ROOT performance benchmarking

Supervisors: O. Shadura, E. Guiraud

L. Harutyunyan

American University of Armenia

Harutyunyan

Table of Content

1. Rootbench: what and why?
2. New benchmarks in rootbench.git
3. Flamegraphs
4. Improvements in RBSupport library
5. Future Items

Rootbench: what and why?

Continuous performance monitoring system for ROOT

Rich and customizable visualizations and aids for performance analysis

https://github.com/root-project/rootbench

Rootbench: what and why?

Usually based on **gbenchmark** micro benchmarking infrastructure

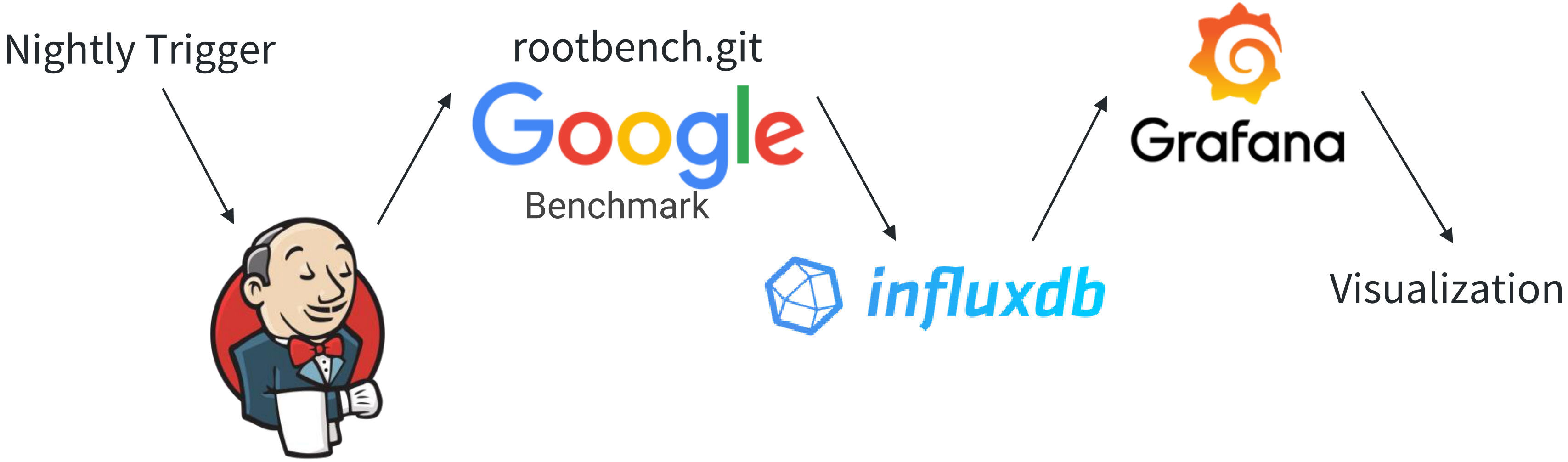
Micro-benchmarks focus on a single function or small piece of functionality

Useful for monitoring performance of software hotspots

https://github.com/google/benchmark

Rootbench: what and why?

Technology



Nightly Trigger

rootbench.git

Benchmark

Visualization

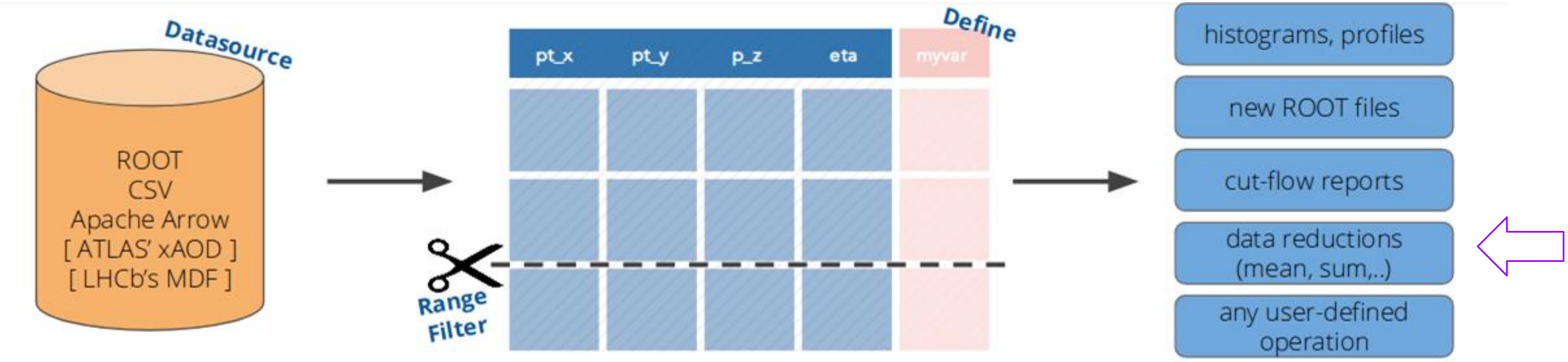
Rootbench: what and why?

In performance analysis of ROOT it is mostly interesting to see

- Wall clock time
- User CPU time
- Memory measurements (implemented only for interpreter benchmarks and for interpreted PyROOT)
- Stack traces (new feature)

New 26 benchmarks in rootbench.git

RDataFrame

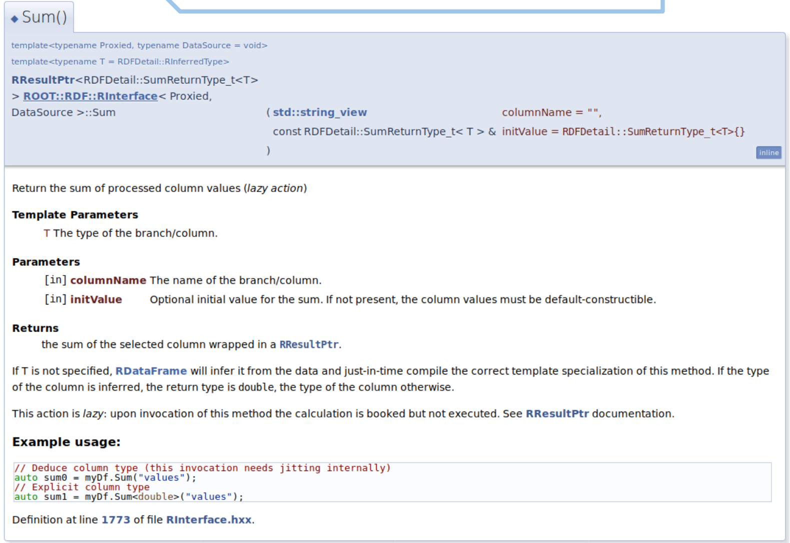


https://indico.cern.ch/event/587955/contributions/2937534/

New 14 RDataFrame benchmarks

- BM_RDataFrameSum_CreateDFFromTFileAndBookSum
- BM_RDataFrameSum_SumScalarDF
- BM_RDataFrameSum_SumScalarWithForeach
- BM_RDataFrameSum_SumScalarAfter**X**Defines
- BM_RDataFrameSum_SumVectorAfter**X**Defines
- ..and others

RDF::Sum()



◆ Sum()

```
template<typename Proxied, typename DataSource = void>
template<typename T = RDFDetail::RInferredType>
RResultPtr<RDFDetail::SumReturnType_t<T>
> ROOT::RDF::RInterface< Proxied,
DataSource >::Sum                ( std::string_view                          columnName = "",
                                   const RDFDetail::SumReturnType_t< T > &  initValue = RDFDetail::SumReturnType_t<T>{}
                                 )
```

inline

Return the sum of processed column values (*lazy action*)

Template Parameters

T The type of the branch/column.

Parameters

[in] **columnName** The name of the branch/column.

[in] **initValue** Optional initial value for the sum. If not present, the column values must be default-constructible.

Returns

the sum of the selected column wrapped in a RResultPtr.

If T is not specified, RDataFrame will infer it from the data and just-in-time compile the correct template specialization of this method. If the type of the column is inferred, the return type is double, the type of the column otherwise.

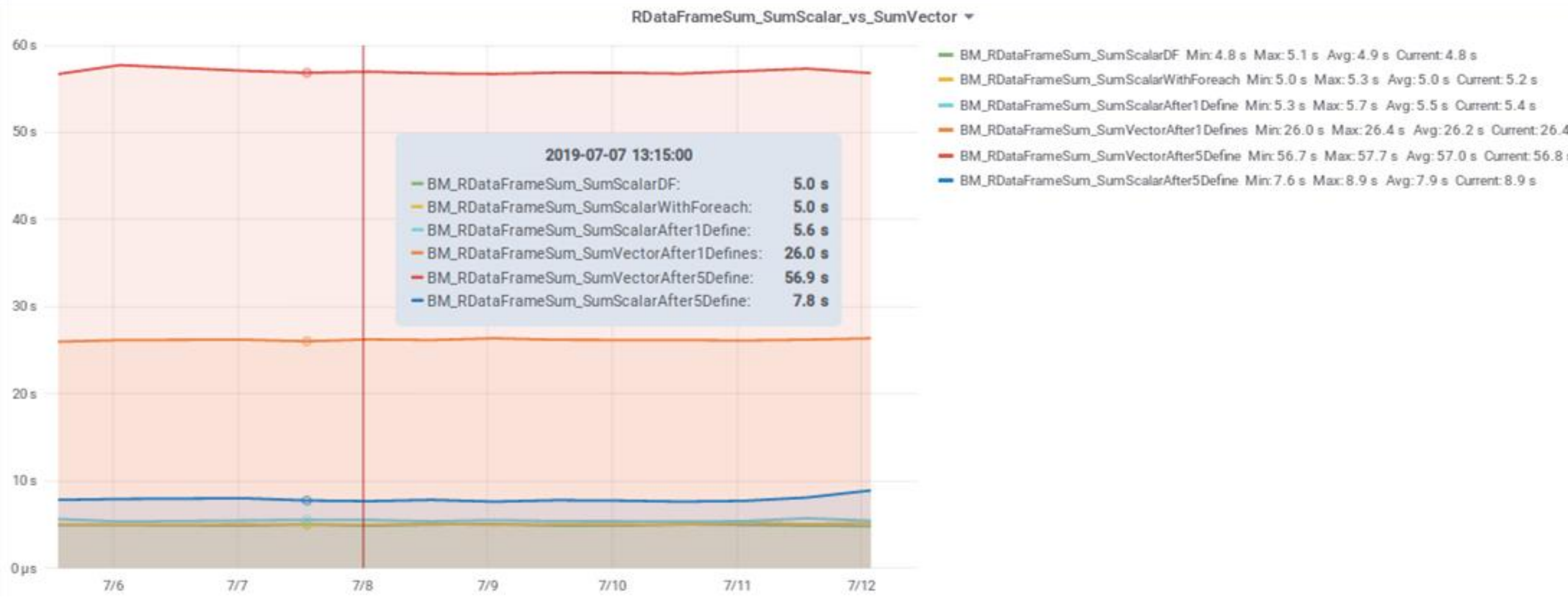
This action is *lazy*: upon invocation of this method the calculation is booked but not executed. See RResultPtr documentation.

Example usage:

```
// Deduce column type (this invocation needs jitting internally)
auto sum0 = myDf.Sum("values");
// Explicit column type
auto sum1 = myDf.Sum<double>("values");
```

Definition at line 1773 of file RInterface.hxx.

New 14 RDataFrame Benchmarks



RDataFrameSum_SumScalar_vs_SumVector

- BM_RDataFrameSum_SumScalarDF Min: 4.8 s Max: 5.1 s Avg: 4.9 s Current: 4.8 s
- BM_RDataFrameSum_SumScalarWithForeach Min: 5.0 s Max: 5.3 s Avg: 5.0 s Current: 5.2 s
- BM_RDataFrameSum_SumScalarAfter1Define Min: 5.3 s Max: 5.7 s Avg: 5.5 s Current: 5.4 s
- BM_RDataFrameSum_SumVectorAfter1Defines Min: 26.0 s Max: 26.4 s Avg: 26.2 s Current: 26.4
- BM_RDataFrameSum_SumVectorAfter5Define Min: 56.7 s Max: 57.7 s Avg: 57.0 s Current: 56.8
- BM_RDataFrameSum_SumScalarAfter5Define Min: 7.6 s Max: 8.9 s Avg: 7.9 s Current: 8.9 s

2019-07-07 13:15:00

BM_RDataFrameSum_SumScalarDF:	5.0 s
BM_RDataFrameSum_SumScalarWithForeach:	5.0 s
BM_RDataFrameSum_SumScalarAfter1Define:	5.6 s
BM_RDataFrameSum_SumVectorAfter1Defines:	26.0 s
BM_RDataFrameSum_SumVectorAfter5Define:	56.9 s
BM_RDataFrameSum_SumScalarAfter5Define:	7.8 s

New TBranch & TTreeReader benchmarks

- BM_TTreePlayer_FixedSizeArrayTBranch
- BM_TTreePlayer_VarSizeArrayTBranch
- BM_TTreePlayer_StdVectorArrayTBranch
- BM_TTreePlayer_FixedSizeArrayReaderArray
- BM_TTreePlayer_VarSizeArrayReaderArray
- BM_TTreePlayer_StdVectorReaderArray
- ..and others

TTree:: GetEntry(Long64_t event..)
vs
TTreePlayer::Next()

We benchmarked
two different types of ROOT Tree event loops readers!

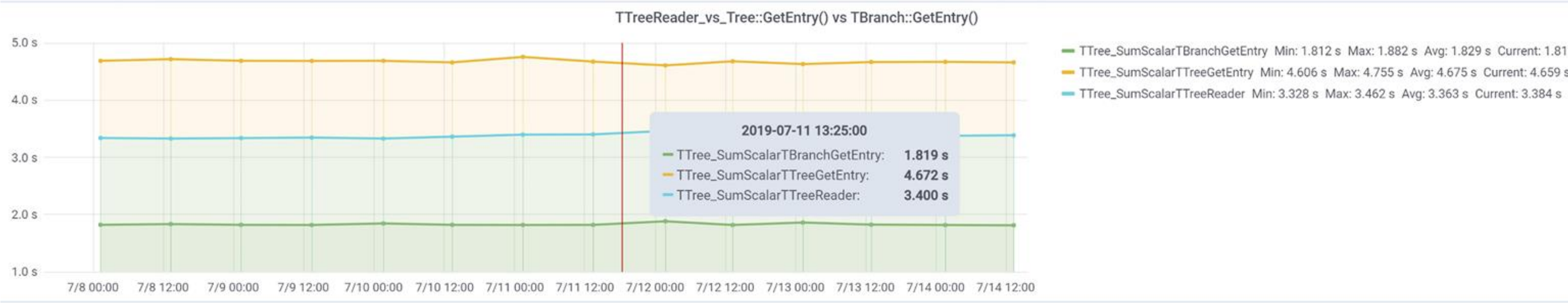
1.https://root.cern.ch/doc/master/classTTree.html#a9fc48df5560fce1a2d63ecd1ac5b40cb
2.https://root.cern.ch/doc/master/classTTreeReader.html#af7b3aa2ea7b5b9a54b3aed57bab4d0d7

New TBranch:Sum() & TTreeReader::Sum() benchmarks

- BM_TTree_SumScalarTBranchGetEntry
- BM_TTree_SumScalarTTreeGetEntry
- BM_TTree_SumScalarTTreeReader
- BM_TTree_SumVectorTBranchGetEntry
- BM_TTree_SumVectorTTreeReader
- ..and others

RDataFrame::Sum() vs
Summation using TTreeReader
vs
Summation using
TBranch::GetEntry()

We benchmarked
two different ways to do the summation operation both for RDF and ROOT TTree/TreePlayer event loop readers!


TTreeReader_vs_Tree::GetEntry() vs TBranch::GetEntry()
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
7/8 00:00
7/8 12:00
7/9 00:00
7/9 12:00
7/10 00:00
7/10 12:00
7/11 00:00
7/11 12:00
7/12 00:00
7/12 12:00
7/13 00:00
7/13 12:00
7/14 00:00
7/14 12:00
2019-07-11 13:25:00
TTree_SumScalarTBranchGetEntry: 1.819 s
TTree_SumScalarTTreeGetEntry: 4.672 s
TTree_SumScalarTTreeReader: 3.400 s
TTree_SumScalarTBranchGetEntry Min: 1.812 s Max: 1.882 s Avg: 1.829 s Current: 1.81
TTree_SumScalarTTreeGetEntry Min: 4.606 s Max: 4.755 s Avg: 4.675 s Current: 4.659 s
TTree_SumScalarTTreeReader Min: 3.328 s Max: 3.462 s Avg: 3.363 s Current: 3.384 s

Performance analysis: Flame Graphs

http://www.brendangregg.com/flamegraphs.html

Flame Graphs: what and why?

Interactive visualizations of profiled software.

Most frequent code-paths are identified quickly and accurately.

Different types of flame graphs:

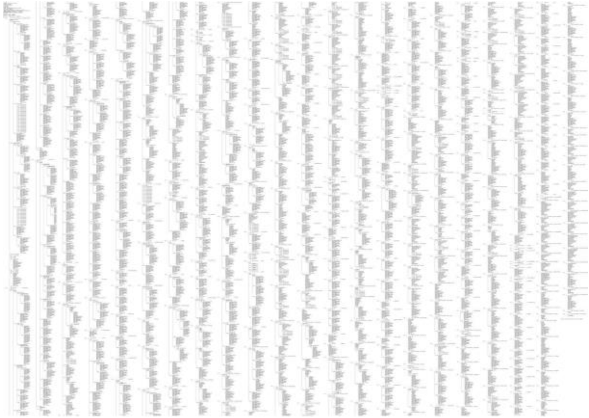
CPU Memory Off-CPU Hot/Cold Differential

Flame Graphs: what and why?

- All data in one picture
- Interactive using JavaScript and browser
- Each box represents a function
- Box width is proportional to the total time a function was profiled directly or its children were profiled

Flame Graphs: Generation

1. Profile event of interest (perf, eBPF, SystemTap, Instruments, DTrace etc.)

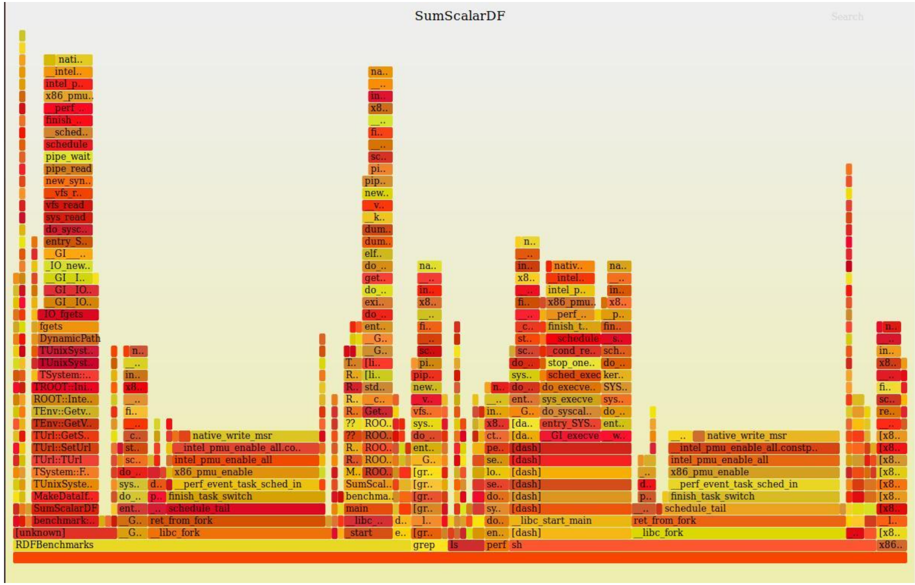


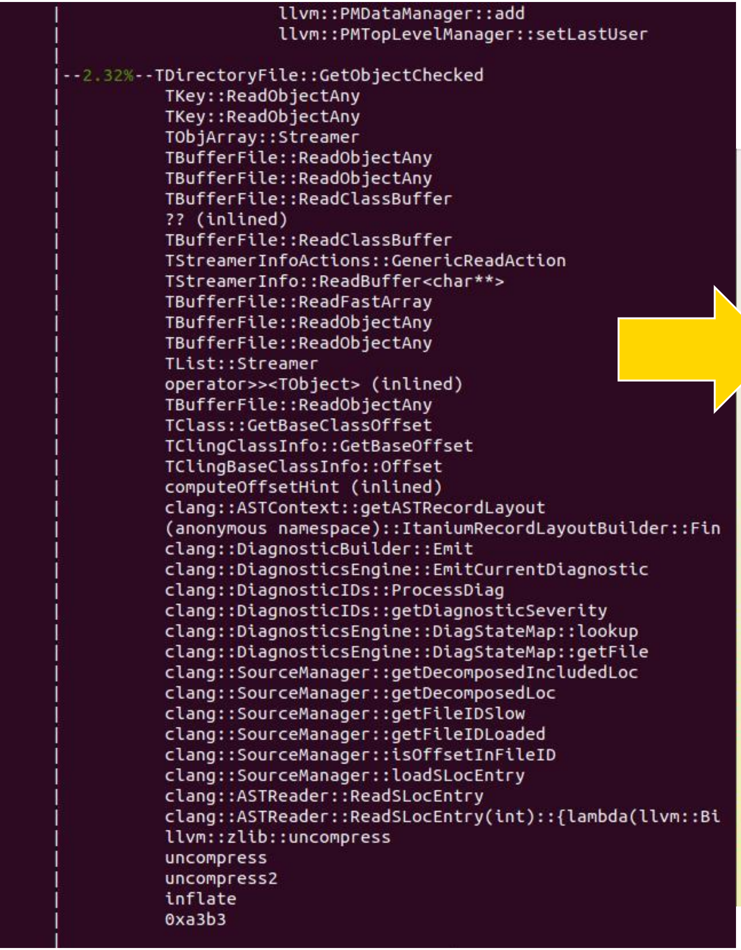
1. Stackcollapse.pl
   - Converts profile data into a single line record.
   - Full output is many lines, one line per stack. Grep can be used to filter stacks before FlameGraphs

Flame Graphs: Generation

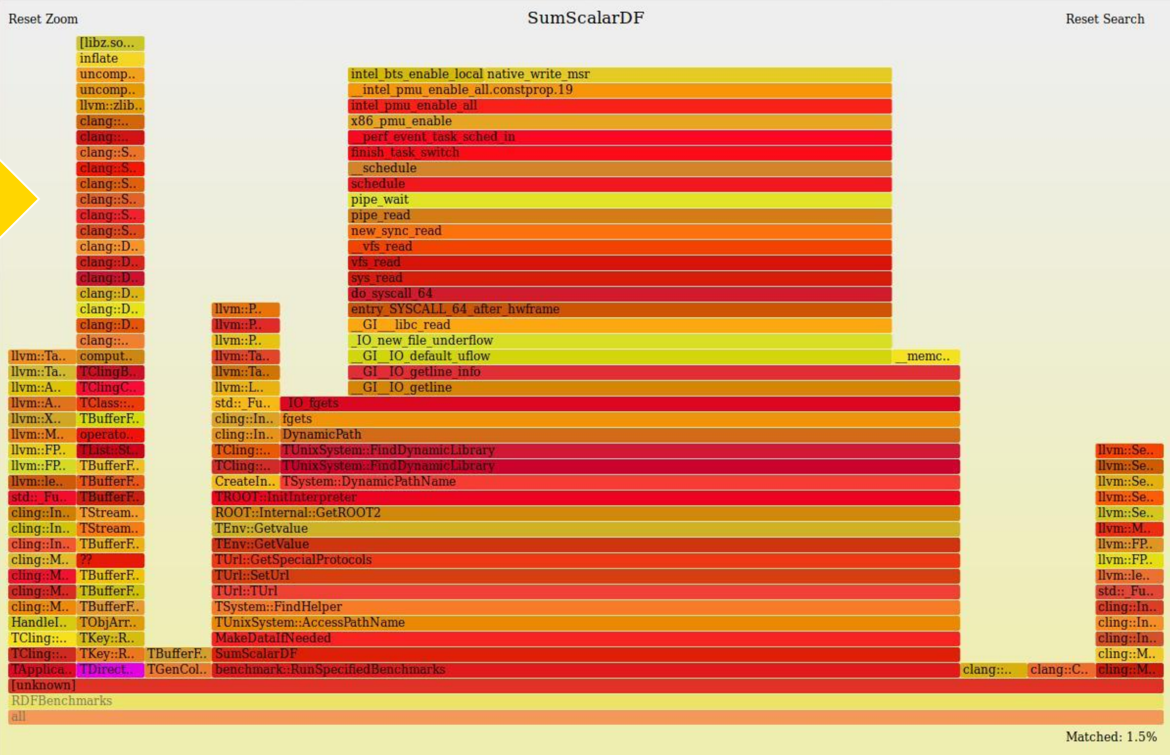
3. flamegraph.pl

   - Converts folded stacks into an interactive SVG





```
 |                  llvm::PMDataManager::add
 |                  llvm::PMTopLevelManager::setLastUser
 |
 |--2.32%--TDirectoryFile::GetObjectChecked
 |          TKey::ReadObjectAny
 |          TKey::ReadObjectAny
 |          TObjArray::Streamer
 |          TBufferFile::ReadObjectAny
 |          TBufferFile::ReadObjectAny
 |          TBufferFile::ReadClassBuffer
 |          ?? (inlined)
 |          TBufferFile::ReadClassBuffer
 |          TStreamerInfoActions::GenericReadAction
 |          TStreamerInfo::ReadBuffer<char**>
 |          TBufferFile::ReadFastArray
 |          TBufferFile::ReadObjectAny
 |          TBufferFile::ReadObjectAny
 |          TList::Streamer
 |          operator>><TObject> (inlined)
 |          TBufferFile::ReadObjectAny
 |          TClass::GetBaseClassOffset
 |          TClingClassInfo::GetBaseOffset
 |          TClingBaseClassInfo::Offset
 |          computeOffsetHint (inlined)
 |          clang::ASTContext::getASTRecordLayout
 |          (anonymous namespace)::ItaniumRecordLayoutBuilder::Fin
 |          clang::DiagnosticBuilder::Emit
 |          clang::DiagnosticsEngine::EmitCurrentDiagnostic
 |          clang::DiagnosticIDs::ProcessDiag
 |          clang::DiagnosticIDs::getDiagnosticSeverity
 |          clang::DiagnosticsEngine::DiagStateMap::lookup
 |          clang::DiagnosticsEngine::DiagStateMap::getFile
 |          clang::SourceManager::getDecomposedIncludedLoc
 |          clang::SourceManager::getDecomposedLoc
 |          clang::SourceManager::getFileIDSlow
 |          clang::SourceManager::getFileIDLoaded
 |          clang::SourceManager::isOffsetInFileID
 |          clang::SourceManager::loadSLocEntry
 |          clang::ASTReader::ReadSLocEntry
 |          clang::ASTReader::ReadSLocEntry(int)::{lambda(llvm::Bi
 |          llvm::zlib::uncompress
 |          uncompress
 |          uncompress2
 |          inflate
 |          0xa3b3
 |
```

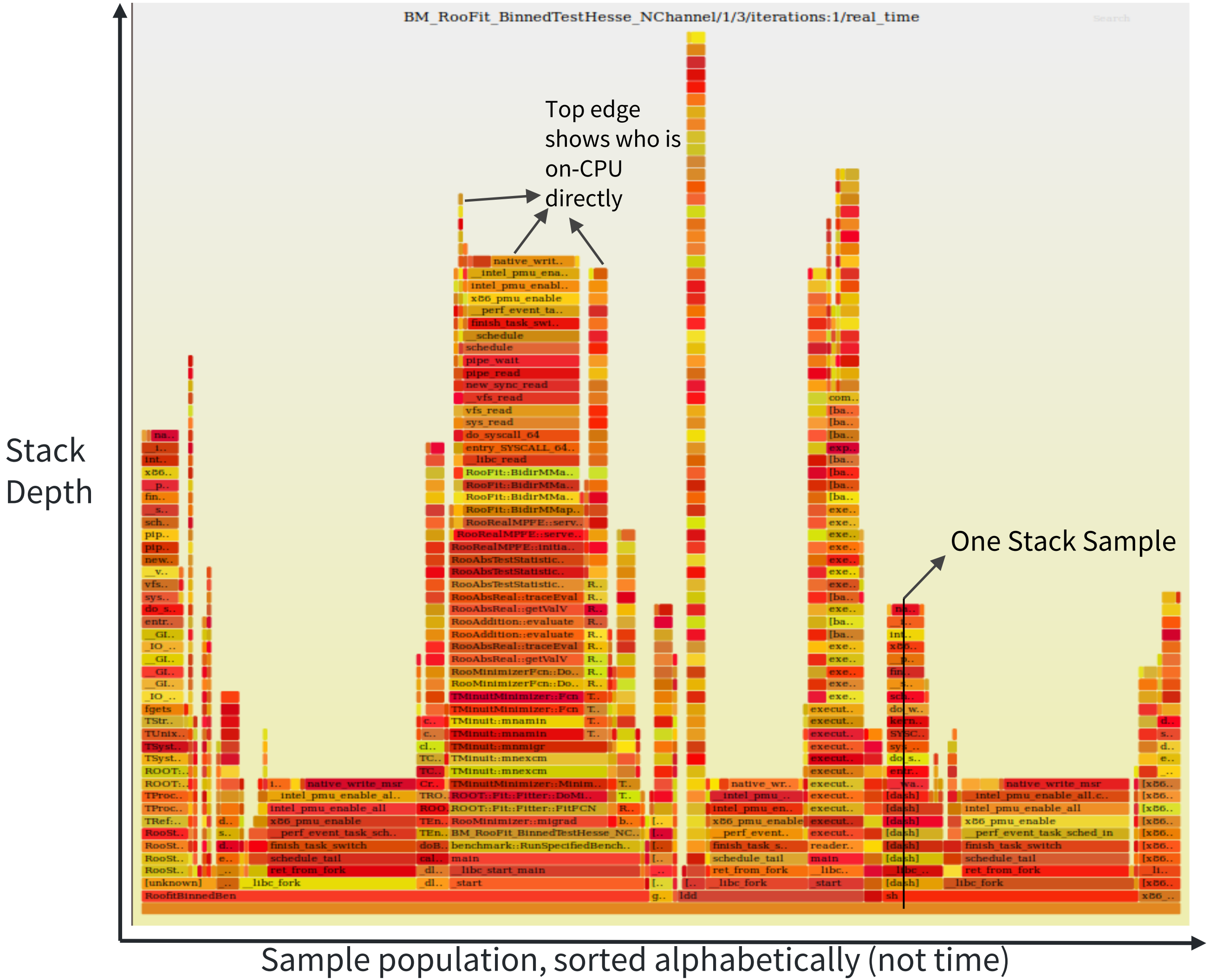


FlameGraphs: CPU

Measure code paths that consume CPU

Understand and optimize CPU usage, improving performance and scalability

Performed by sampling CPU stack traces at a timed interval

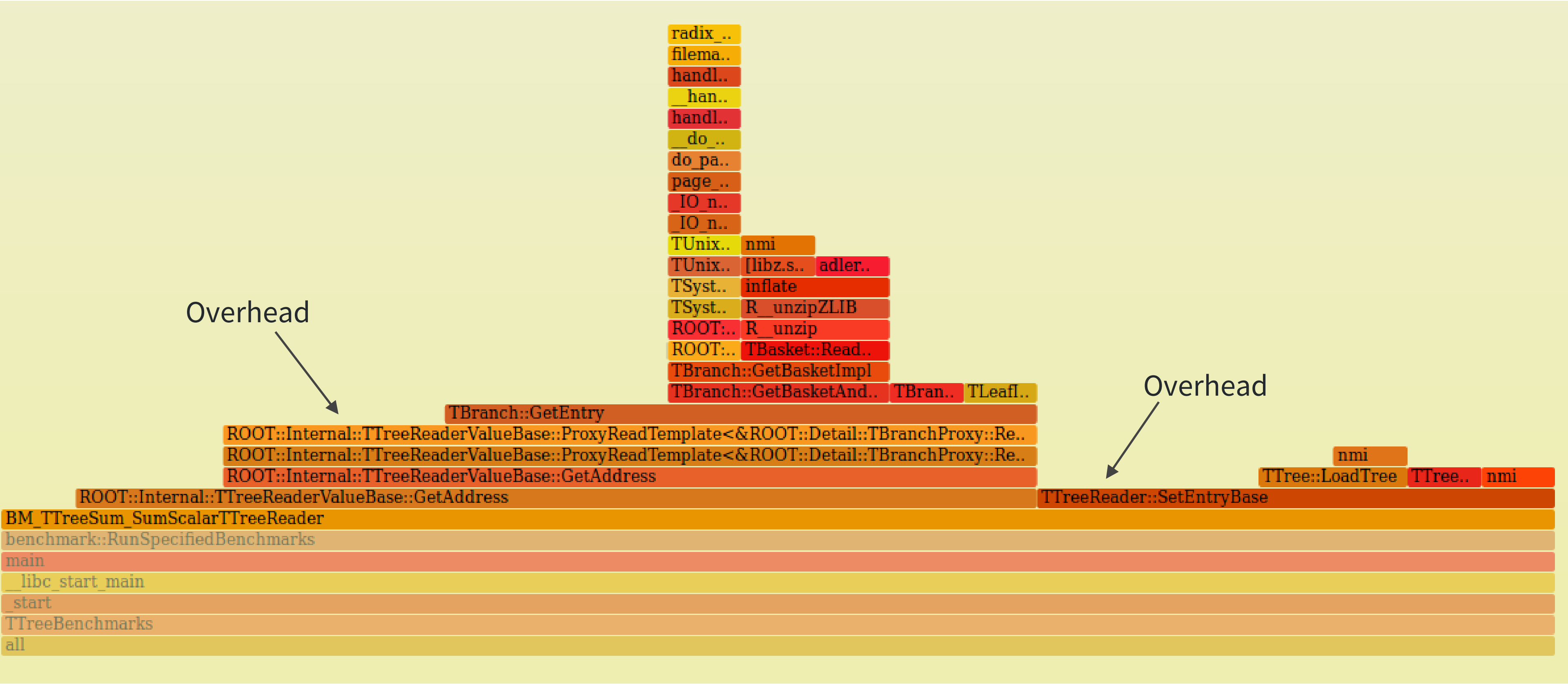


Stack Depth

Top edge shows who is on-CPU directly

One Stack Sample

Sample population, sorted alphabetically (not time)



Overhead

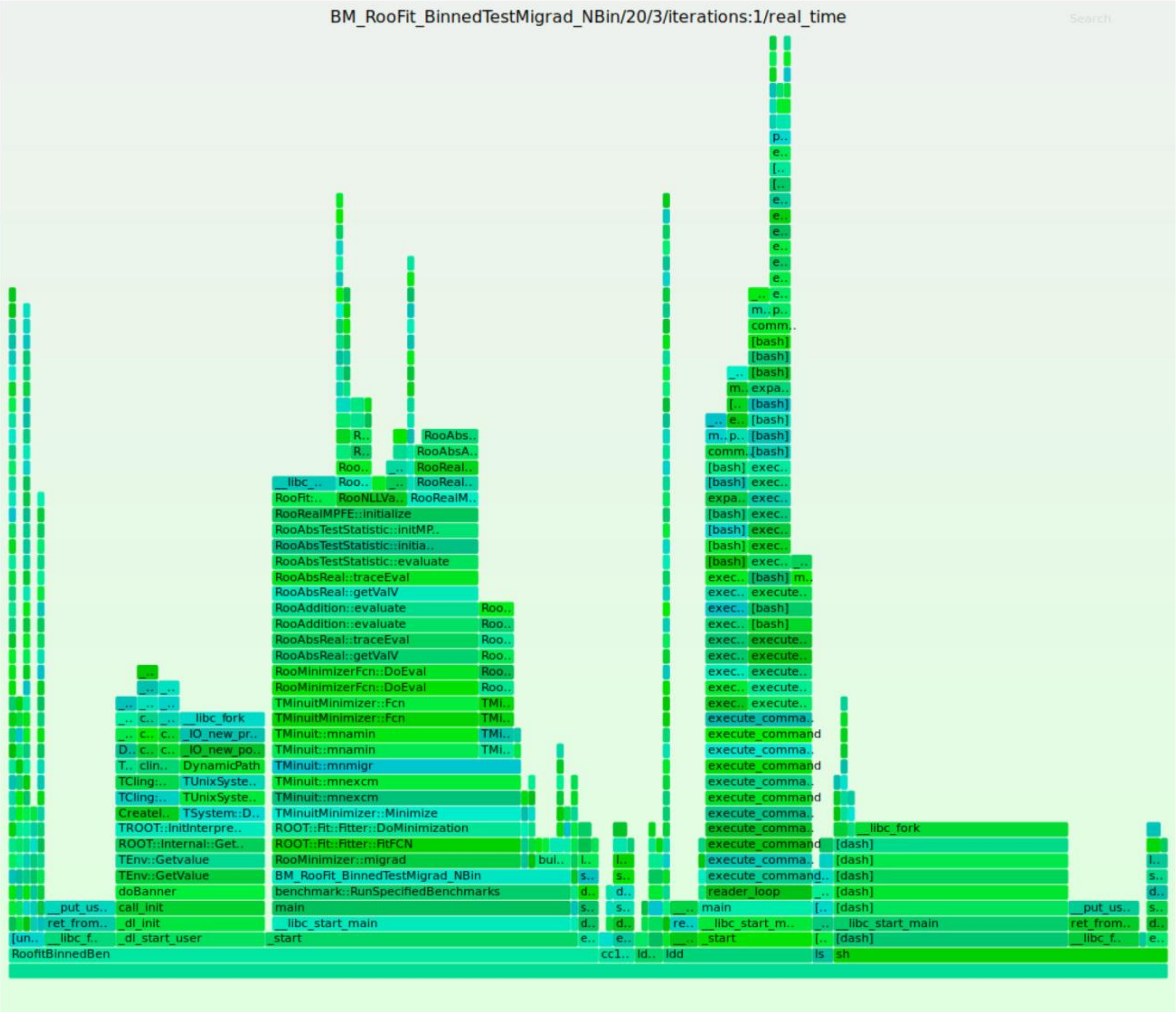
Overhead

FlameGraphs: Memory

Memory FlameGraphs are helpful when analyzing memory growth or leaks

by tracing one of the following memory events:

- Allocator functions: malloc(), free()
- brk() syscall
- mmap() syscall
- Page faults


BM_RooFit_BinnedTestMigrad_NBin/20/3/iterations:1/real_time
Search
RooAbs..
RooAbsA..
RooReal..
RooReal..
RooRealM..
RooFit:..
RooNLLVa..
__libc_..
RooRealMPFE::initialize
RooAbsTestStatistic::initMP..
RooAbsTestStatistic::initia..
RooAbsTestStatistic::evaluate
RooAbsReal::traceEval
RooAbsReal::getValV
RooAddition::evaluate
RooAddition::evaluate
RooAbsReal::traceEval
RooAbsReal::getValV
RooMinimizerFcn::DoEval
RooMinimizerFcn::DoEval
TMinuitMinimizer::Fcn
TMinuitMinimizer::Fcn
TMinuit::mnamin
TMinuit::mnamin
TMinuit::mnmigr
TMinuit::mnexcm
TMinuit::mnexcm
TMinuitMinimizer::Minimize
ROOT::Fit::Fitter::DoMinimization
ROOT::Fit::Fitter::FitFCN
RooMinimizer::migrad
BM_RooFit_BinnedTestMigrad_NBin
benchmark::RunSpecifiedBenchmarks
main
__libc_start_main
_start
__libc_fork
_IO_new_pr..
_IO_new_po..
DynamicPath
TUnixSyste..
TUnixSyste..
TSystem::D..
TCling:..
TCling:..
CreateI..
TROOT::InitInterpre..
ROOT::Internal::Get..
TEnv::Getvalue
TEnv::GetValue
doBanner
call_init
_dl_init
_dl_start_user
__put_us..
ret_from..
__libc_f..
[un..
RoofitBinnedBen
cc1..
ld..
ldd
[bash]
exec..
execute..
execute_comma..
execute_command
reader_loop
main
__libc_start_m..
_start
ls
__libc_fork
[dash]
sh

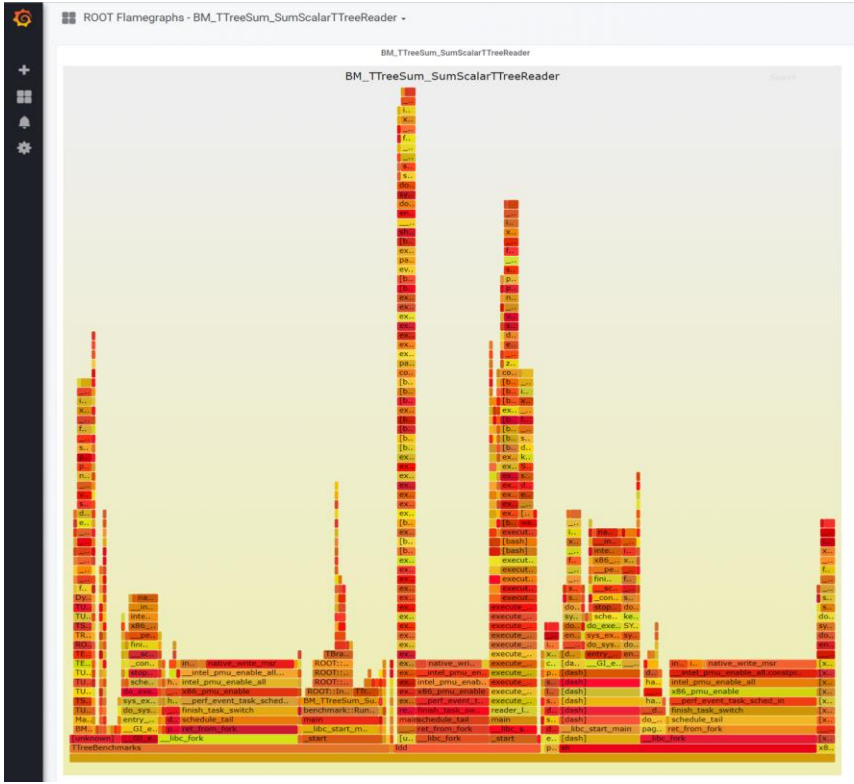
How to display generated FlameGraphs

- Generate them locally

https://github.com/HLilit/rootbench/tree/flamegraph-lilit/rootbench-scripts

- Download automatically generated flamegraphs directly from page
- Grafana SVG Panel (simple to store/display, work in progress to upload automatically)

TTreeSum_SumScalarTBranchGetEntry, TTreeSum_SumScalarTTreeGetEntry, TTreeSum_SumScalarTTreeReader



Improvements in RBSupport library

Added memory measurements for interpreted PyROOT and improved the memory measurements for ROOT interpreter.

https://github.com/root-project/rootbench/pull/95

Future Items

- Finish porting iotools.git (IO benchmarks from ACAT 2017 presented by Jakob Blomer) into rootbench.git

- Finalize PRs about flamegraphs and improved memory measurements in rootbench.git

https://github.com/root-project/rootbench/pull/94
https://github.com/root-project/rootbench/pull/95

Summer Student project final report - https://cds.cern.ch/record/2684053?ln=en

Thank you!