

Enhancements to ROOT performance benchmarking

Supervisors: O. Shadura, E. Guiraud

L. Harutyunyan

American University of Armenia



Table of Content

1. Rootbench: what and why?
2. New benchmarks in rootbench.git
3. Flamegraphs
4. Plans for next 2 weeks

Rootbench: what and why?

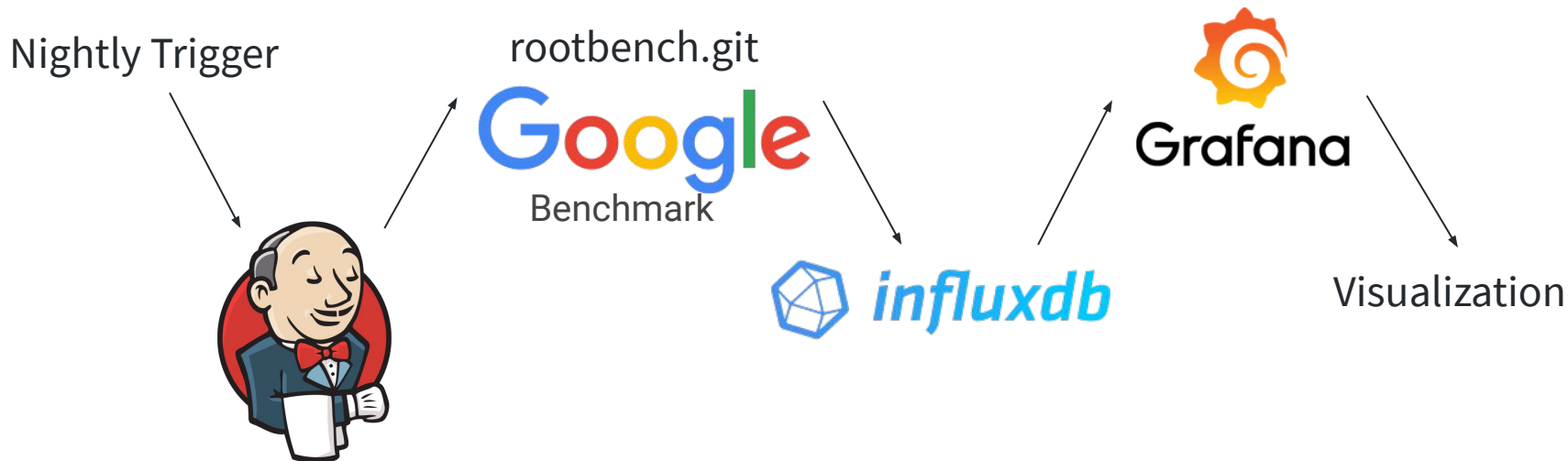
Continuous performance monitoring system for ROOT

Rich and customizable visualizations and aids for performance analysis

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

Rootbench: what and why?

Technology



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>

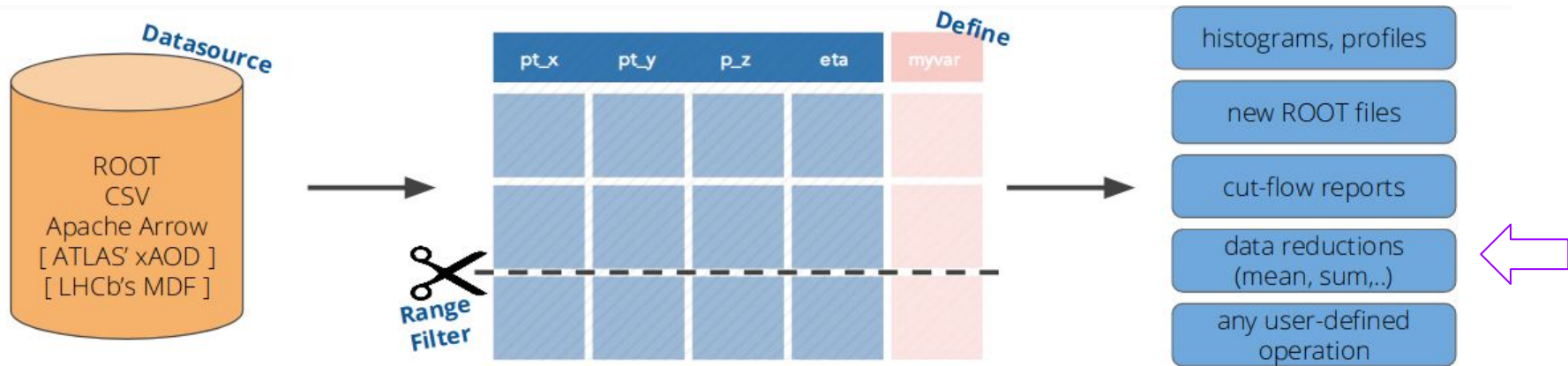
Rootbench: what and why?

In performance analysis of ROOT it is mostly interesting to see

- Wall clock time
- User CPU time
- Memory measurements (implemented only for interpreter benchmarks)
- 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_CreateDFFromT
FileAndBookSum
- BM_RDataFrameSum_SumScalarDF
- BM_RDataFrameSum_SumScalarWith
Foreach
- BM_RDataFrameSum_SumScalarAfter
XDefines
- BM_RDataFrameSum_SumVectorAfter
XDefines
- ..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>{}
)

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.

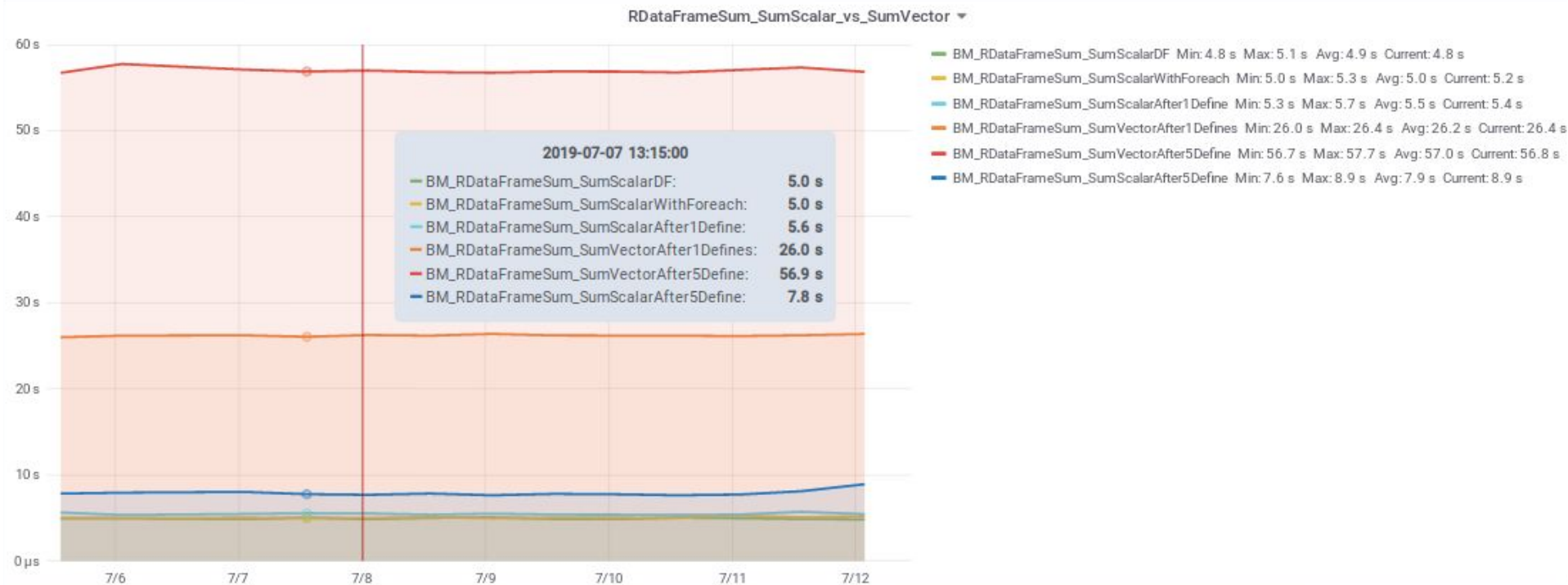
If T is not specified, RDataFrame will infer it from the data and just-in-time compile the correct template specialization of this method. If the type of the column is inferred, the return type is double, the type of the column otherwise.

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



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!

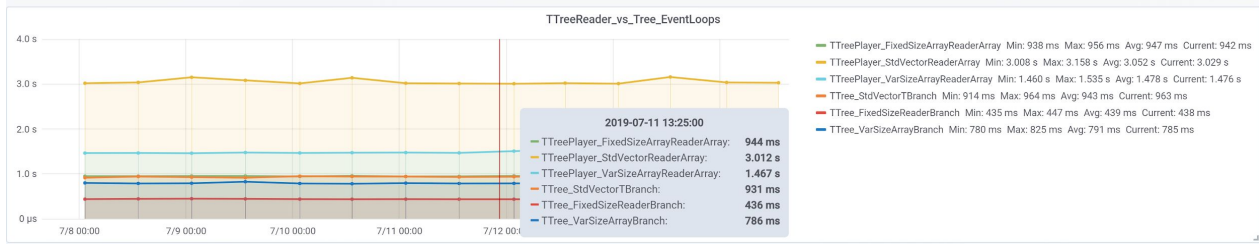
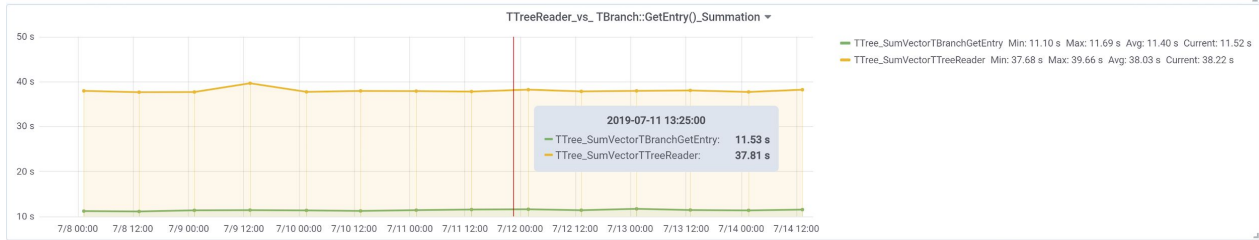
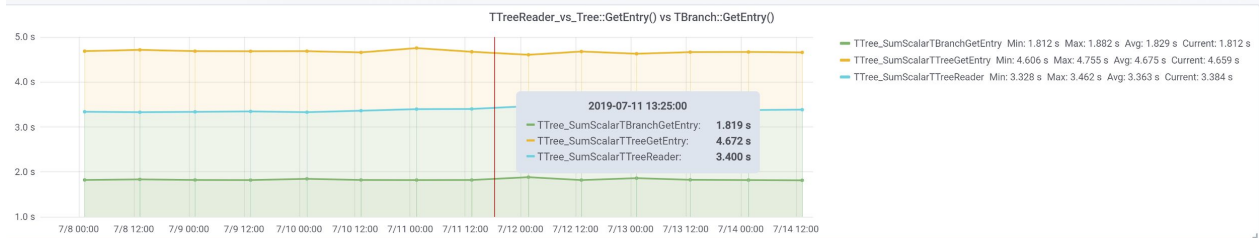
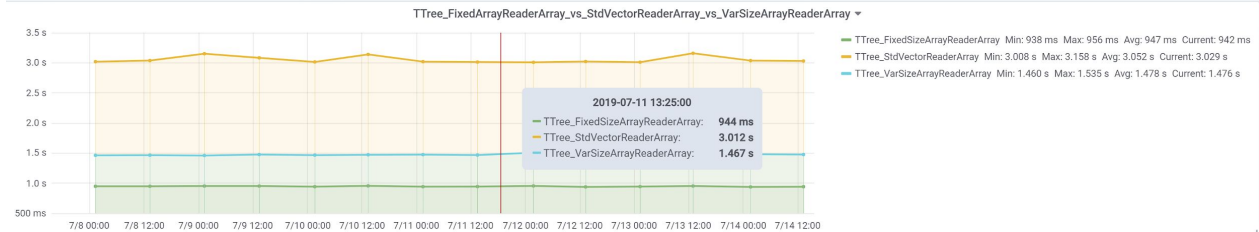
1. <https://root.cern.ch/doc/master/classTTree.html#a9fc48df5560fce1a2d63ecd1ac5b40cb>
2. <https://root.cern.ch/doc/master/classTTreeReader.html#af7b3aa2ea7b5b9a54b3aed57bab4d0d7>

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 way to do the summation
operation both for RDF and ROOT
TTree/TreePlayer event loop readers!



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

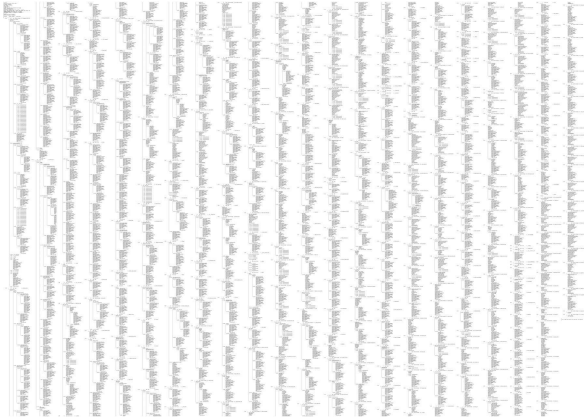


Flame Graphs: what and why?

- 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.)



2. 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



FlameGraphs: CPU

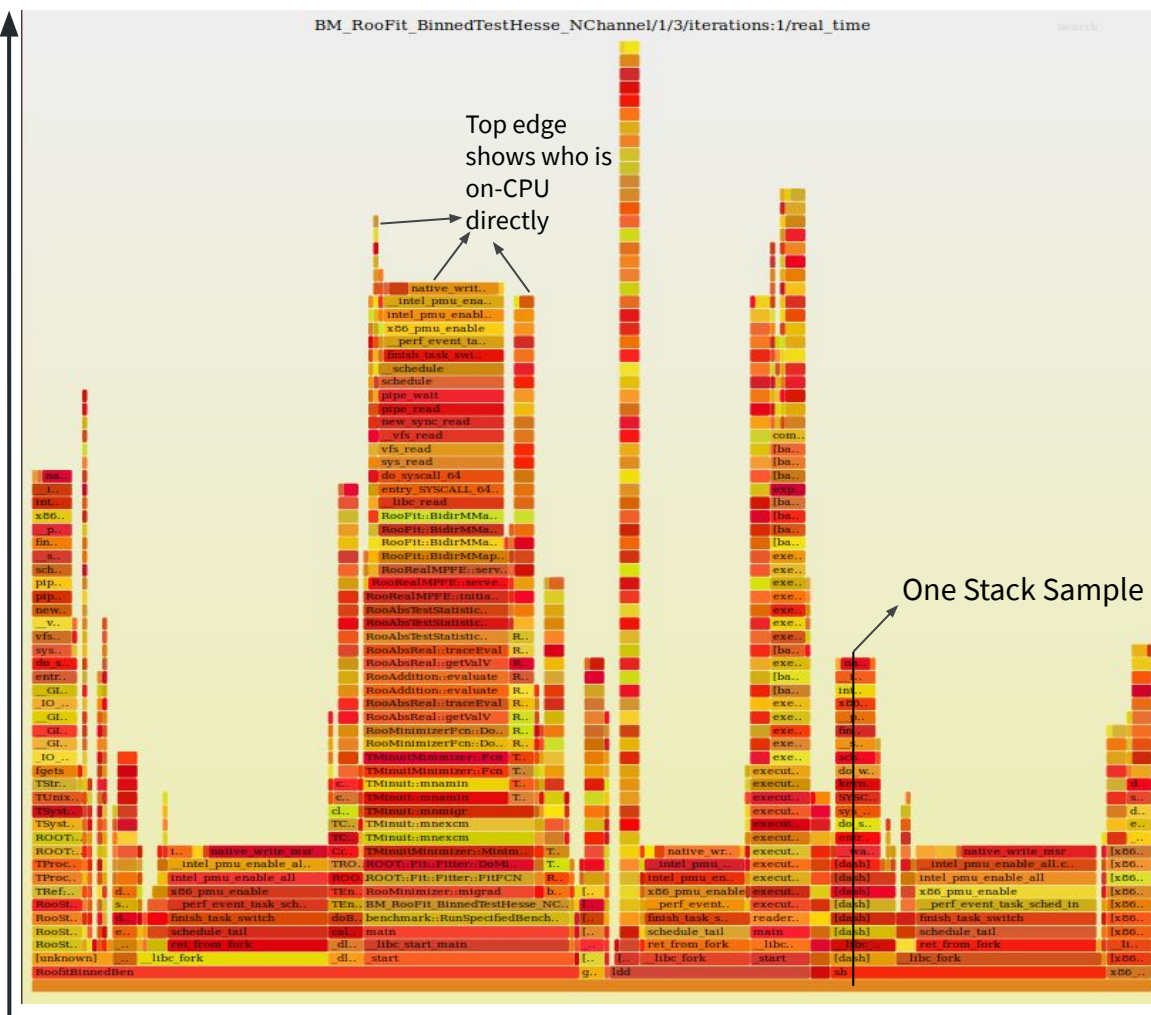
Measure code paths that consume CPU

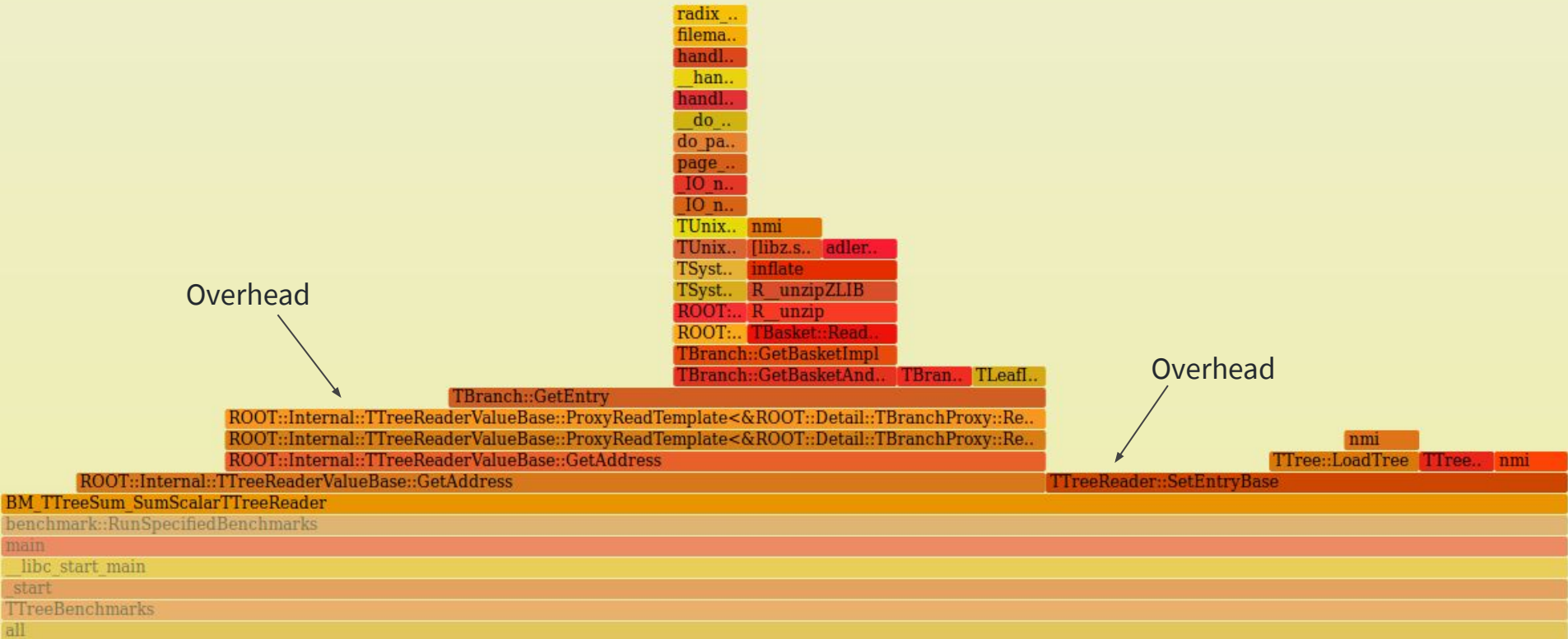
Understand and optimize CPU usage, improving performance and scalability

Performed by sampling CPU stack traces at a timed interval

Code path branches can reveal key functions

Stack Depth





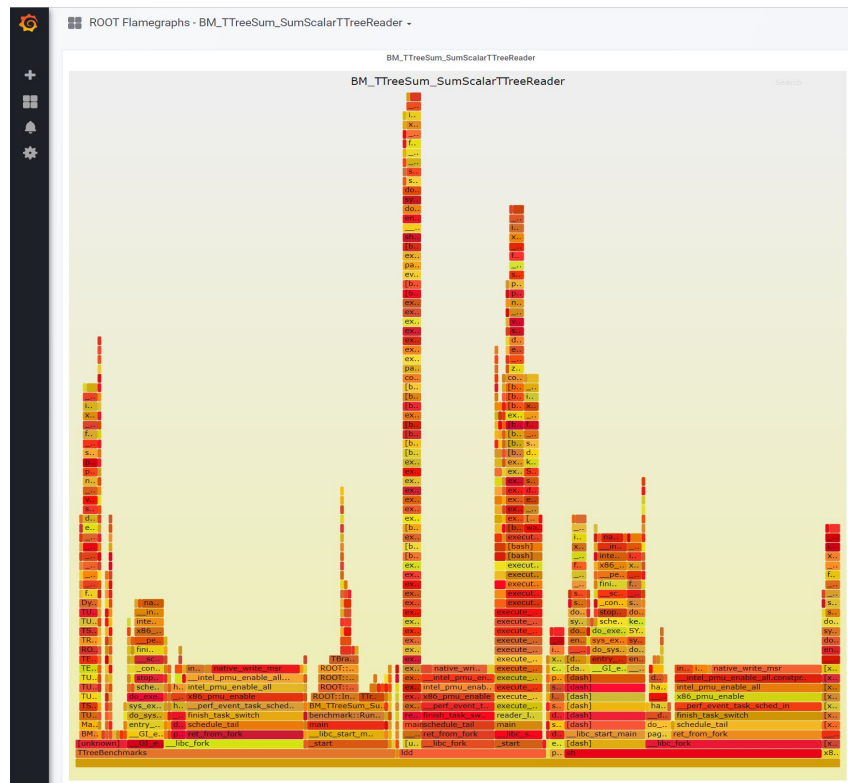
How to display generated FlameGraphs

- 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)

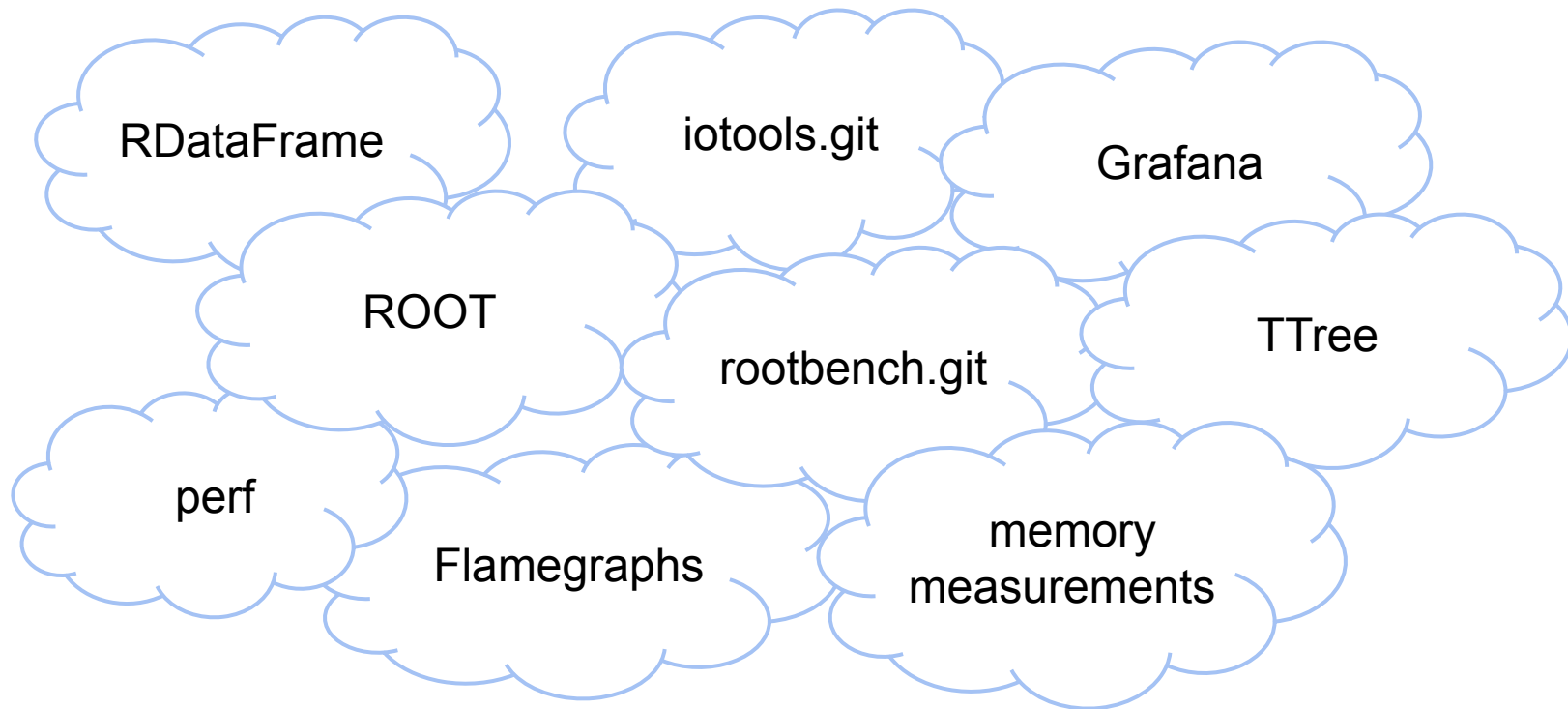
[TTreeSum_SumScalarTBranchGetEntry,](#)
[TTreeSum_SumScalarTTreeGetEntry,](#)
[TTreeSum_SumScalarTTreeReader](#)



Plans for next 2 weeks

- 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>
- Collect feedback from team members

My CERN Summer Student adventures



<https://sft.its.cern.ch/jira/browse/ROOT-10179>

My CERN Summer Student adventures



Thank you, amazing ROOT team!!!
(particularly my supervisors O.Shadura & E.Guiraud and other team members)