Two more tools in your bag

- ➔ debugging C++ template instantiations
- ➔ profiling runtimes with flamegraphs

Enrico Guiraud

Software we love #3 18/08/2017

Every good work of software starts by scratching a developer's personal itch.

The Cathedral and the Bazaar, Eric S. Raymond

Tool #1: motivation

Issue

long compilation times for an integration test (~15s)

Where is the compiler spending time?

~~I know! I will look at the generated assembly!~~

check out Matt Godbolt's compiler exporer!

Compilation time reports

`g++ -ftime-report` (partial output)

```
Execution times (seconds)
 phase setup             :   0.00 ( 0%) usr   0.00 ( 0%) sys   0.01 ( 0%) wall    1339 kB ( 0%)
ggc
 phase parsing           :   0.70 (13%) usr   0.20 (32%) sys   0.90 (15%) wall  176271 kB (27%)
ggc
 phase lang. deferred    :   1.87 (35%) usr   0.13 (21%) sys   2.01 (34%) wall  210148 kB (32%)
ggc
 phase opt and generate  :   2.73 (52%) usr   0.29 (47%) sys   3.00 (51%) wall  259294 kB (40%)
ggc
 phase finalize          :   0.00 ( 0%) usr   0.00 ( 0%) sys   0.01 ( 0%) wall       0 kB ( 0%)
ggc
 preprocessing           :   0.07 ( 1%) usr   0.05 ( 8%) sys   0.12 ( 2%) wall    4969 kB ( 1%)
ggc
 parser (global)         :   0.17 ( 3%) usr   0.07 (11%) sys   0.23 ( 4%) wall   69082 kB (11%)
ggc
 parser struct body      :   0.17 ( 3%) usr   0.03 ( 5%) sys   0.15 ( 3%) wall   25272 kB ( 4%)
ggc
 parser function body    :   0.03 ( 1%) usr   0.00 ( 0%) sys   0.05 ( 1%) wall    6656 kB ( 1%)
```

Compilation time reports

`g++ -ftime-report` (partial output)

```
template instantiation  :   2.25 (38%) wall
```

Templight: a compiler profiler

Clang-based tool to profile the time and memory consumption of template instantiations

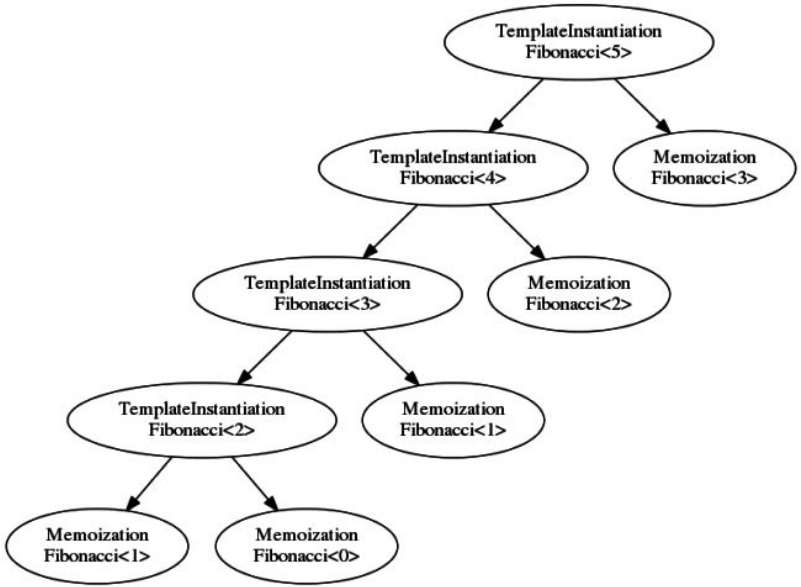
```
template <unsigned int N>
struct Fibonacci {
  static const unsigned int value =
    Fibonacci<N-1>::value + Fibonacci<N-2>::value;
};

template <>
struct Fibonacci<0> { static const unsigned int value = 0; };

template <>
struct Fibonacci<1> { static const unsigned int value = 1; };

int main() {
  return Fibonacci<5>::value;
};
```

“Flat” profile



Templight: a compiler profiler

Clang-based tool to profile the time and memory consumption of template instantiations

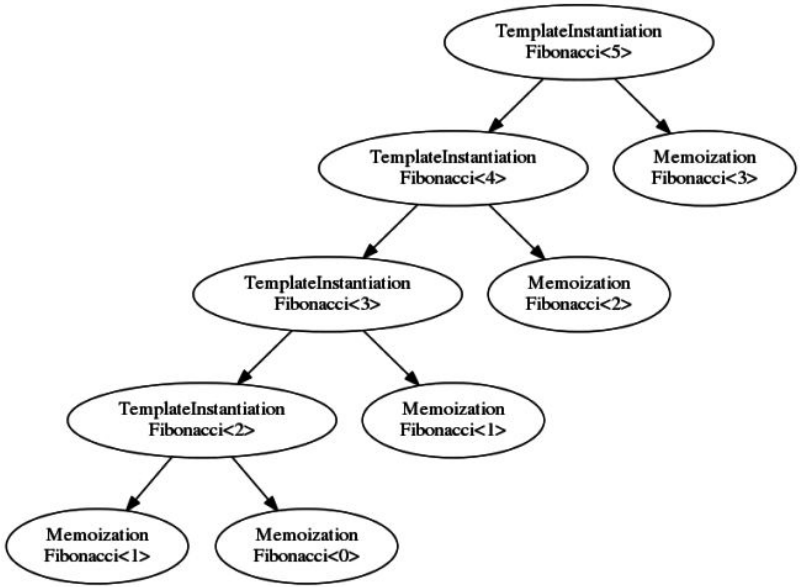
```
template <unsigned int N>
struct Fibonacci {
  static const unsigned int value =
    Fibonacci<N-1>::value + Fibonacci<N-2>::value;
};

template <>
struct Fibonacci<0> { static const unsigned int value = 0; };

template <>
struct Fibonacci<1> { static const unsigned int value = 1; };

int main() {
  return Fibonacci<5>::value;
};
```

“Flat” profile



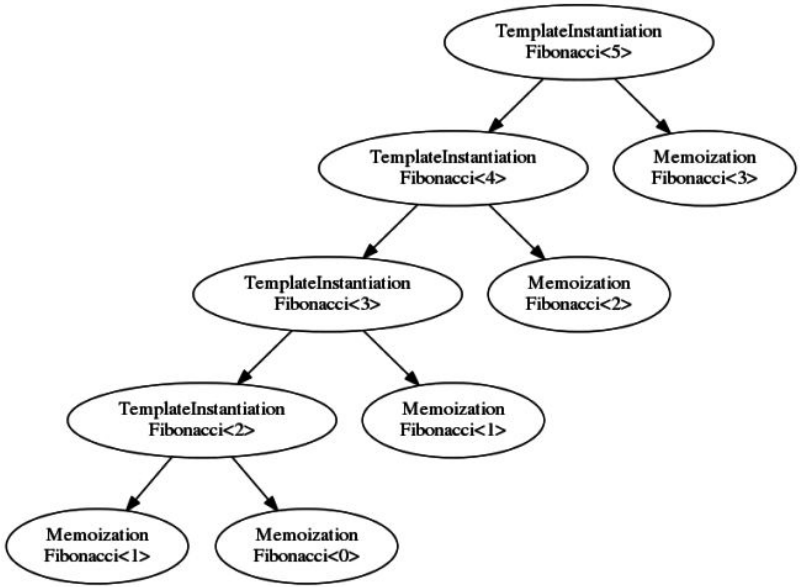
Templight: a compiler profiler

Clang-based tool to profile the time and memory consumption of template instantiations

```
template <unsigned int n>
constexpr int fib()
  if constexpr (n < 2)
      return n;
  else
      return fib<n-1>() + fib<n-2>();
};

int main() {
  return fib<5>();
};
```

“Flat” profile



Get yourself a templight

Installing Templight
requires to be compiled as a Clang tool

1) fetch LLVM and Clang's sources
2) clone Templight into clang/tools
3) compile everything together

templight and templight-tools on Github

Get yourself a templight

dvzubarev commented on Jan 16

For anyone who wants to build templight at the present moment.
You need to use `0738fa1` version of templight and 289544 revision of llvm/clang.

👍 2

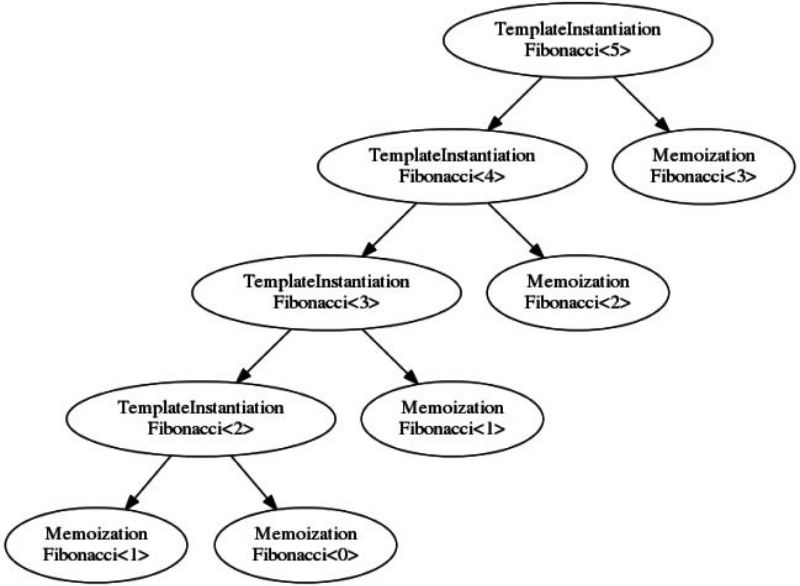
bluehood commented on Feb 2

thank you.

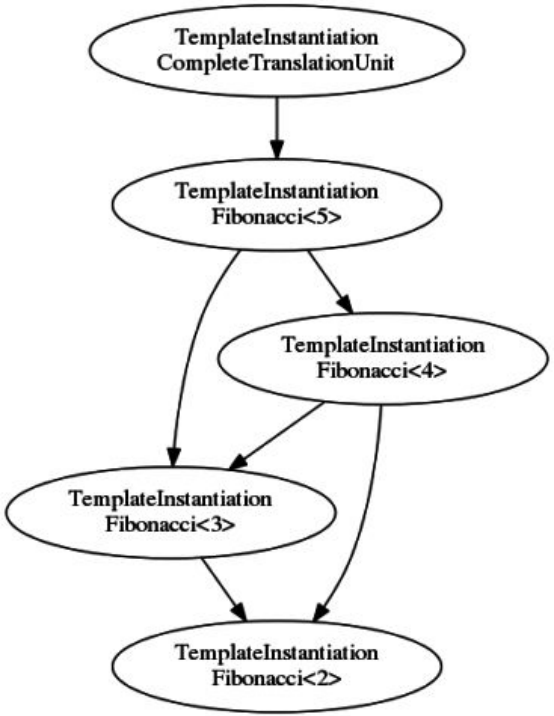
Templight: a compiler profiler

Flat vs call-graph profiles

“Flat” profile

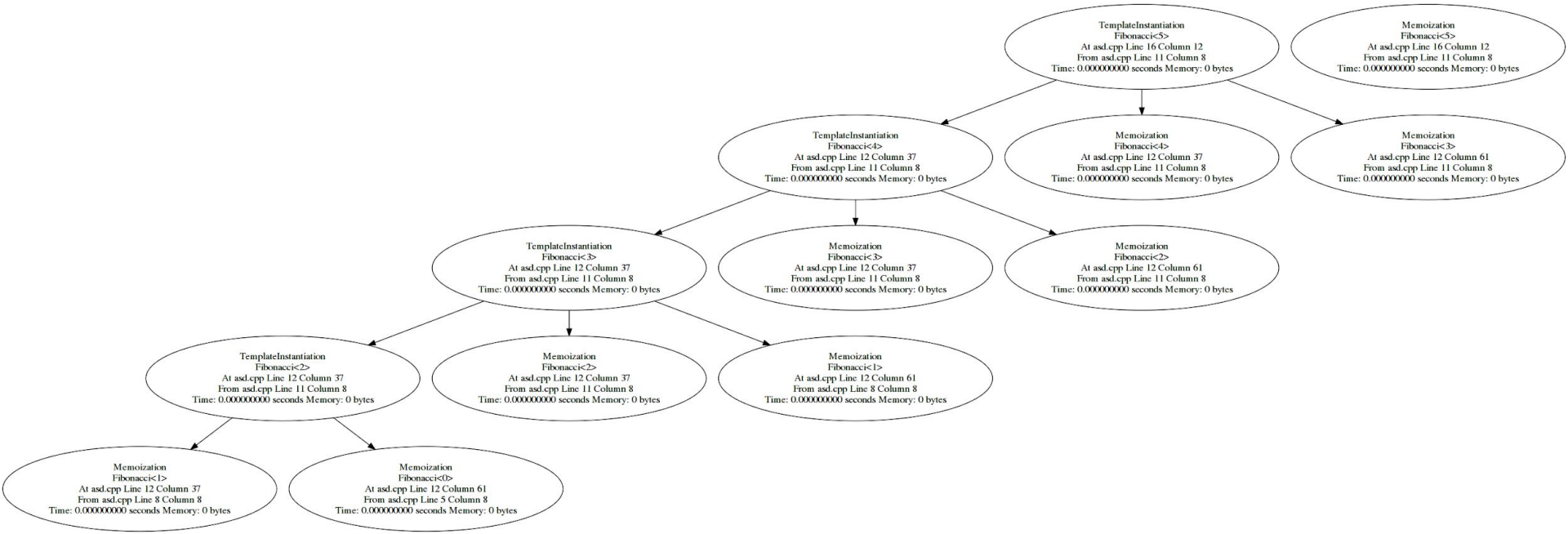


“Call-graph” profile



Templight: a compiler profiler

...vs reality: fibonacci



Templight: a compiler profiler

...vs reality:one of my tests

I had to leave graphviz behind in favour of yEd, a java applet for graph visualization

Good ol' wall of text

```
TemplateInstantiation CompleteTranslationUnit  6.635999943
0.480000000 TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDa
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>
TemplateInstantiation std::make_shared<Operations::FillTOOperation,std::shared_ptr<TH1F>&,unsignedint&>  0.128000000
TemplateInstantiation std::make_shared<TH1F,TH1F&>  0.116000000
TemplateInstantiation std::make_shared<Operations::FillOperation,std::shared_ptr<TH1F>&,unsignedint&>  0.115999999
TemplateInstantiation std::make_shared<bool,bool>  0.108000000
TemplateInstantiation std::vector<std::shared_ptr<bool>,std::allocator<std::shared_ptr<bool>>>::emplace_back<std::shared_ptr<bool>&>  0.103999999
TemplateInstantiation std::unordered_set<std::basic_string<char>,std::hash<string>,std::equal_to<std::basic_string<char>>,std::allocator<std::basic_s
TemplateInstantiation std::vector<double,std::allocator<double>>::emplace_back<constdouble&>  0.091999999
TemplateInstantiation std::unique_ptr<TH1F,std::default_delete<TH1F>>  0.060000000
TemplateInstantiation std::map<std::thread::id,unsignedint,std::less<std::thread::id>,std::allocator<std::pair<conststd::thread::id,unsignedint>>>  0
TemplateInstantiation std::vector<std::shared_ptr<TDataFrameActionBase>,std::allocator<std::shared_ptr<ROOT::Internal::TDataFrameActionBase>>>::~vect
TemplateInstantiation std::map<std::basic_string<char>,std::shared_ptr<TDataFrameBranchBase>,std::less<std::basic_string<char>>,std::allocator<std::p
TemplateInstantiation std::vector<std::basic_string<char>,std::allocator<std::basic_string<char>>>::vector  0.043999999
TemplateInstantiation std::vector<std::pair<longlong,TBranch*>,std::allocator<std::pair<longlong,TBranch*>>>  0.040000000
TemplateInstantiation std::vector<TSchemaHelper,std::allocator<ROOT::Internal::TSchemaHelper>>  0.036000000
TemplateInstantiation std::vector<std::shared_ptr<TDataFrameFilterBase>,std::allocator<std::shared_ptr<ROOT::Detail::TDataFrameFilterBase>>>::~vector
TemplateInstantiation std::vector<longlong,std::allocator<longlong>>  0.035999999
```

Reduced number of shared_ptrs

```
TemplateInstantiation CompleteTranslationUnit  3.135999948
0.463999999 TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:739:31),TDataFrameImpl>,(lam
TemplateInstantiation std::make_shared<TDataFrameAction<(lambdaat/home/eguiraud/ROOT/root_install/include/ROOT/TDataFrame.hxx:744:31),TDataFrameImpl>,(lambdaat/home/e
TemplateInstantiation std::make_shared<Operations::FillTOOperation,std::shared_ptr<TH1F>&,unsignedint&>  0.123999999
TemplateInstantiation std::make_shared<TH1F,TH1F&>  0.108000000
TemplateInstantiation std::make_shared<Operations::FillOperation,std::shared_ptr<TH1F>&,unsignedint&>  0.108000000
TemplateInstantiation std::make_shared<bool,bool>  0.099999999
TemplateInstantiation std::vector<std::shared_ptr<bool>,std::allocator<std::shared_ptr<bool>>>::emplace_back<std::shared_ptr<bool>&>  0.096000000
TemplateInstantiation std::unordered_set<std::basic_string<char>,std::hash<string>,std::equal_to<std::basic_string<char>>,std::allocator<std::basic_string<char>>>  0.
TemplateInstantiation std::unique_ptr<TH1F,std::default_delete<TH1F>>  0.060000000
TemplateInstantiation std::map<std::thread::id,unsignedint,std::less<std::thread::id>,std::allocator<std::pair<conststd::thread::id,unsignedint>>>  0.052000000
TemplateInstantiation std::vector<std::basic_string<char>,std::allocator<std::basic_string<char>>>::vector  0.044000000
TemplateInstantiation std::map<std::basic_string<char>,std::shared_ptr<TDataFrameBranchBase>,std::less<std::basic_string<char>>,std::allocator<std::pair<conststd::bas
TemplateInstantiation std::vector<TSchemaHelper,std::allocator<ROOT::Internal::TSchemaHelper>>  0.036000000
TemplateInstantiation std::vector<longlong,std::allocator<longlong>>  0.036000000
TemplateInstantiation std::vector<std::shared_ptr<TDataFrameActionBase>,std::allocator<std::shared_ptr<ROOT::Internal::TDataFrameActionBase>>>::~vector  0.035999999
TemplateInstantiation std::vector<std::pair<longlong,TBranch*>,std::allocator<std::pair<longlong,TBranch*>>>  0.035999999
TemplateInstantiation std::vector<std::basic_string<char>,std::allocator<std::basic_string<char>>>::vector  0.035999999
TemplateInstantiation std::vector<unsignedlong,std::allocator<unsignedlong>>  0.032000000
TemplateInstantiation std::vector<unsignedlonglong,std::allocator<unsignedlonglong>>  0.032000000
TemplateInstantiation std::vector<TVirtualArray*,std::allocator<TVirtualArray*>>  0.032000000
TemplateInstantiation std::vector<TDirectory*,std::allocator<TDirectory*>>  0.032000000
```

Templight: bottomline

- timings can be hard to interpret
- difference in call-graph and flat profiles is not clear (to me)
- not exactly a drop-in tool

- gives some insight on which templates are costly
- only tool out there which does what it does

(templight can also work as an interactive “debugging” console
that steps through template instantiations - never tried)

C++ runtime profiling

You can see a lot by just looking.

Yogi Berra, baseball coach

Standard tools

gprof
requires special compilation options
does not support multi-threading out of the box
does not support profiling of shared libraries out of the box

valgrind --tool=callgrind
crazy slowdown

See "How profilers lie" for a write-up of more subtle issues with gprof and callgrind

perf: your new standard tool

perf's perks

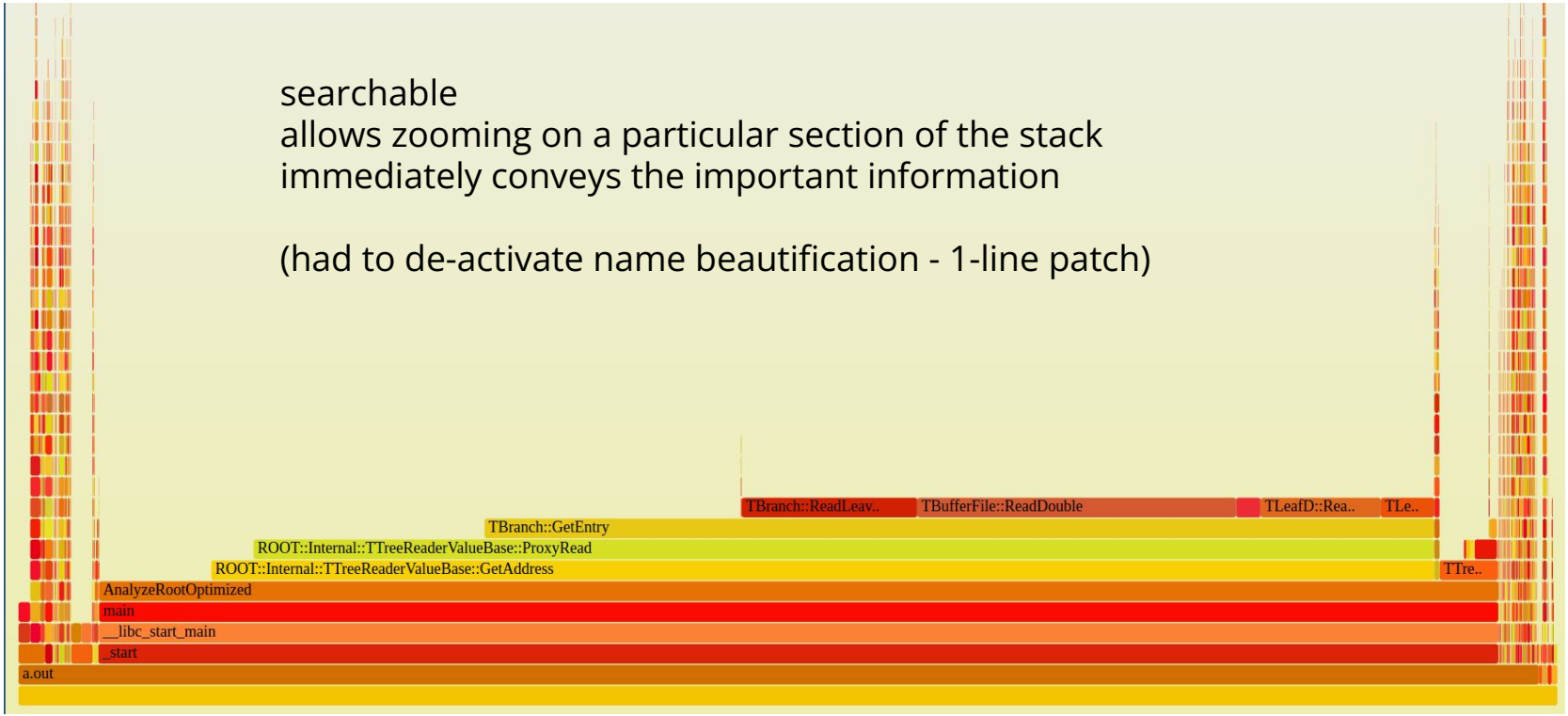
Performance analyzing tool designed for Linux
Available from kernel version 2.6.31
Command line interface
Statistical profiling of the entire system (both kernel and userland code)

As easy as

```
perf record ./main
perf report
```

see the excellent
CppCon 2015: Chandler Carruth "Tuning C++: Benchmarks, and CPUs, and Compilers! Oh My!"

flamegraphs for visualization



[FlameGraph on Github]

CLI cheat sheet

```
perf record --call-graph dwarf "./$PNAME"
perf record -g "./$PNAME"
```

You may not have permission to collect system-wide stats.
Consider tweaking /proc/sys/kernel/perf_event_paranoid:

```
 -1 - Not paranoid at all
  0 - Disallow raw tracepoint access for unpriv
  1 - Disallow cpu events for unpriv
  2 - Disallow kernel profiling for unpriv
```

Getting flamegraphs out

```
perf script | stackcollapse-perf.pl | flamegraph.pl > "output.svg"

flamegraph.pl --title="foo" --fontsize=16 --minwidth=1 --height=20 --width=1800
```

let's play a bit