Yet another Malloc Profiler (this time based on std::stacktrace)
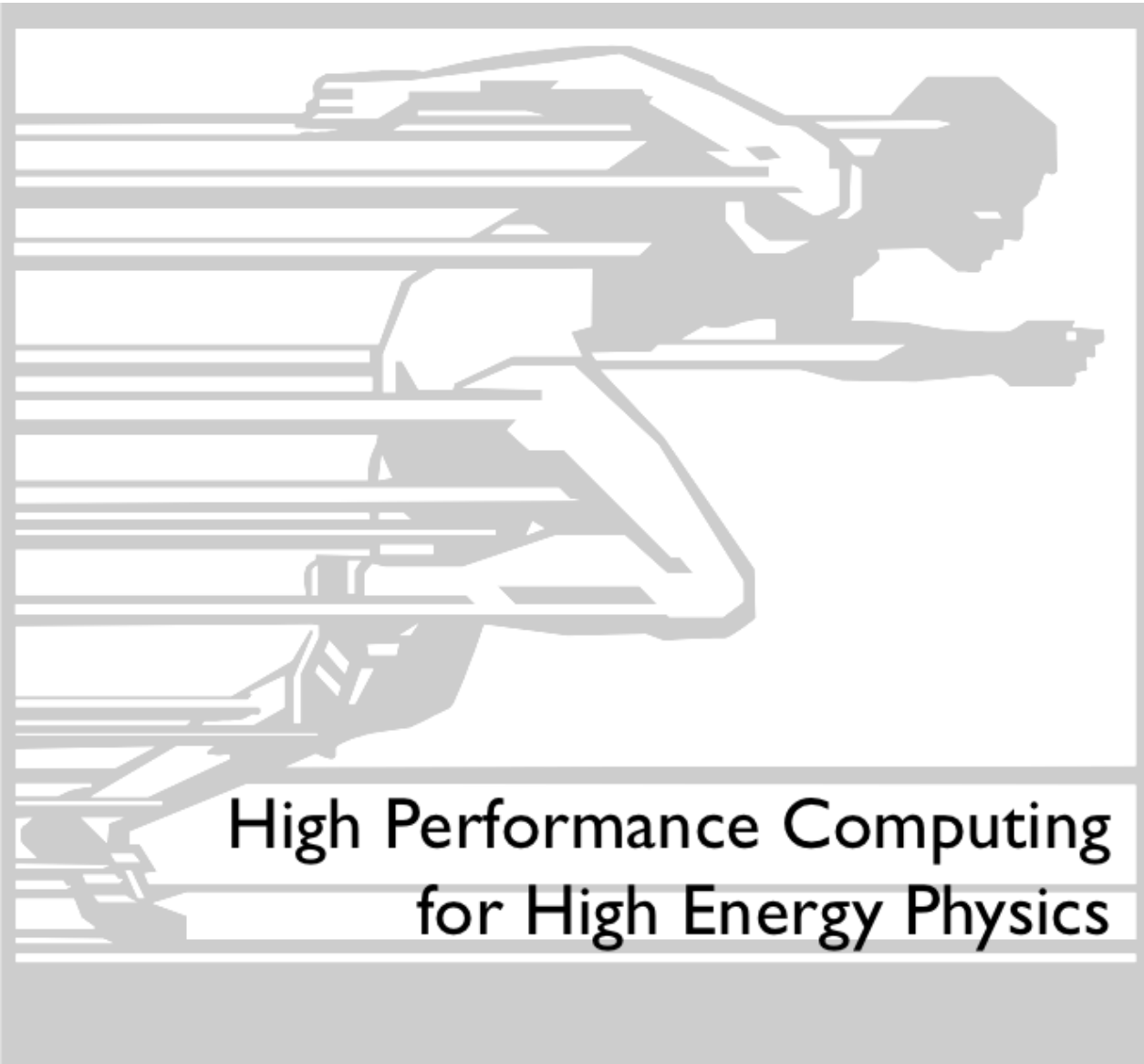
Vincenzo Innnocente

CERN

CMS Experiment

High Performance Computing
for High Energy Physics

Motivations

- Replace igprof (unmaintained)
- Do not use external libraries (they need to be updated, and code ported)
- Keep code simple (easy maintenance)
- Keep output format simple (and possibly human readable)
- Use existing tools for analysis and display

std::backtrace

- In C++23 standard

- "supported" in gcc since v12

- Need compiler to be specifically configured
  - --enable-libstdcxx-backtrace=yes

- Tool needs to be linked with a static library
  - -lstdc++_libbacktrace in gcc12, -lstdc++exp in gcc14

- Three bugs found: fixed in the main branches (last on Nov 15)

112348	gcc	libstdc+	unassigned@gcc.gnu.org	UNCO	---	[C++23] defect in struct hash<basic_stacktrace<_Allocator>>
112263	gcc	libbackt	unassigned@gcc.gnu.org	RESO	FIXE	[C++23] std::stacktrace does not identify symbols in shared library
111936	gcc	libstdc+	redi@gcc.gnu.org	RESO	FIXE	std::stacktrace cannot be used in a shared library

Resources:
https://stackoverflow.com/questions/3899870/how-to-print-a-stack-trace-whenever-a-certain-function-is-called/54365144

Instrumentation

- As everybody else does
  - LD_PRELOAD
  - dlsym(RTLD_NEXT, "malloc"); etc
  - std::stacktrace objects kept in an unordered_map
    - One could try to use a Patricia Trie
  - Accumulation by thread (ncalls, memtot, memlive, max-memlive)
  - Final aggregation and symbol-name resolution at dump time
  - Filtering and "Remangling": In process (dump-time) and post processing
  - Result is a file containing ;-separated-stacktraces with a value associated

- https://github.com/VinInn/MallocProfiler/tree/main

Visualization tools

- FlameGraph
  - https://www.brendangregg.com/flamegraphs.html
  - https://github.com/brendangregg/FlameGraph
  - FlameGraph/flamegraph.pl --width 2400 step3_DumpDoEvent.md > /tmp/step3_DumpDoEvent.svg
- Speedscope (thanks to Giulio Eulisse for pointing it out)
  - https://github.com/jlfwong/speedscope
  - Just open the web-app and drop the file
- Just try
  - scp lxplus8.cern.ch:/afs/cern.ch/user/i/innocent/public/step3_DumpDoEvent.md.gz .
- Screen shots in Backup

caveats

- Cling (via llvm) seems to "interact" with gcc backtrace data structures:
  - Hang deep in gcc_src/libgcc/unwind-dw2-btree.h
  - Solution: overload two functions and stop malloc-recording in there
- tls calls realloc
  - Not tracking realloc at the moment...
- Gcc unwind calls dl_iterate_phdr that seems to be protected by a mutex
  - Stacktrace recording the facto serializes
- Processing 100-event reco-relval takes ~2 hours (with recording threshold of 128 bytes)
- Dump (includes symbol name resolution) is very slow (several minutes)

Remangling

- C++ symbol names can be long (boost::spirit? 30Kbytes!)
- Many stacktrace details are not of interest for the analysis in hand
  - Everything below TClass, TFormula, std::regex and boost::spirit for instance
  - Everything before tbb::detail::d1::function_task
- Many names can be shortened
  - string, vector, etc
  - , allocator<…> can be removed
- The whole signature is often not of interest (function name is enough)

Filtering

- The file is simple text: best post-processing filtering is grep and sed!

[innocent@patatrack01 11634.0_TTbar_14TeV+2021]$
wc step3_DumpNew.mdr
  1,114,944 117815162 **3,354,955,155**
 grep -ic cling step3_DumpNew.mdr
163,652
grep -ic TClass step3_DumpNew.mdr
138,481
grep -ic TFormula step3_DumpNew.mdr
36,210
grep -ic regex step3_DumpNew.mdr
16,827
grep -ic spirit step3_DumpNew.mdr
322,480
grep doEvent step3_DumpNew.mdr | wc
 290,234 28882728 **888,851,647**
after symbol filtering (see next slide)
wc step3_DumpDoEvent.md
  290,194   580388 **333,669,635**

Example of post processing

```
#  3 is memtot, 4 is memlive
grep '_mpTrace_' $1.mdr | sed 's/\;\{1,\}/;/g' | sed 's/^;//' | cut -f1,4 -d'$' \
| sed 's/_mpTrace_;_start;__libc_start_main;main;main::{lambda()#1}::operator();tbb::detail::r1::task_arena_impl::execute;
tbb::detail::d1::task_arena_function<main::{lambda()#1}::operator()() const::{lambda()#1}, void>::operator();//' \
| sed 's/_mpTrace.*tbb::detail::d1::function_task/function_task/' \
| sed 's/std::__cxx11::basic_string<char, std::char_traits<char> >/string/g' \
| sed 's/std::basic_string_view<char, std::char_traits<char> >/string_view/g' \
| sed 's/operator new;malloc;//g' \
| sed 's/malloc;//g' \
| sed 's/unsigned /u/g' | sed 's/operator/op/g' | sed 's/std:://g'
| sed 's/const//g' | sed 's/&//g' | sed 's/ //g' | sed 's/::/:/g' | tr '$' ' ' > $1.md
```

Performance (and some results)

- 100 events, 4 threads, step3 (reco) wf 11634.0_TTbar_14TeV+2021
  - Threshold 128 Bytes (below threshold stacktrace is not recorded)
    - Elapsed time 6950 seconds: (~18 minutes before event 5 starts)
    - Total Number of malloc calls 579,980,716
    - Total MemTot 9.90328e+10 (100GB!)
    - Total Max Memlive 3,929,812,405
    - Raw dump: wc => 1,078,553 116,416,953 3,281,194,659 memdump_2003909_1.mdr
  - SmallAllocations: (<128 Bytes)
    - Number of malloc calls 474,512,716 (80% of tot!)
    - MemTot 1.80925e+10 (18GB)
    - Typical MemLive 524,182,207
  - Inverse Threshold (record stacktrace only for <128 Bytes allocations)
    - 55 events : 18,500 seconds (~2hours before event 5 starts)
    - Raw dump: wc => 2,528,393 213,268,892 6,504,541,718 memdump_99644_15.mdr
    - After filter: wc => 2,532,509   5,065,018 4,002,873,401 memdump_99644_16.md
    - Remove memlive=0 :
      - grep -v "; 0" memdump_99644_16.md | wc => 469,045 938,090 753,865,549

Local Instrumentation


```
diff --git a/RecoTracker/PixelLowPtUtilities/BuildFile.xml b/RecoTracker/PixelLowPtUtilities/BuildFile.xml
index 3226f2c8d56..8350c10d381 100644
--- a/RecoTracker/PixelLowPtUtilities/BuildFile.xml
+++ b/RecoTracker/PixelLowPtUtilities/BuildFile.xml
@@ -14,6 +14,7 @@
 <use name="TrackingTools/PatternTools"/>
 <use name="TrackingTools/Records"/>
 <use name="TrackingTools/TrajectoryFiltering"/>
+<use name="MallocDummy/MallocDummy"/>
 <export>
   <lib name="1"/>
 </export>
```

```
diff --git a/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc b/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc
index 852ff217c4b..1b423e2dc40 100644
--- a/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc
+++ b/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc
@@ -145,6 +145,7 @@ void ClusterShapeHitFilter::fillPixelData() {
   }
 }

+#include "mallocProfiler.h"
 void ClusterShapeHitFilter::fillStripData() {
   // copied from StripCPE (FIXME maybe we should move all this in LocalReco)
   auto const& geom_ = *theTracker;
@@ -158,19 +159,33 @@ void ClusterShapeHitFilter::fillStripData() {
     }
   }

+  std::cout << "in ClusterShapeHitFilter::fillStripData " << dus.size()-offset << ' ' << sizeof(StripData) << ' ' << stripData.max_load_factor() << std::endl;
+  // need to active both
+  mallocProfiler::setThreshold(0);
+  mallocProfiler::activate(mallocProfiler::allThreads);
+  mallocProfiler::activate(mallocProfiler::currentThread);
+
+  stripData.reserve(dus.size()-offset);
   for (auto i = offset; i != dus.size(); ++i) {
     const StripGeomDetUnit* stripdet = (const StripGeomDetUnit*)(dus[i]);
     assert(stripdet->index() == int(i));
     assert(stripdet->type().isTrackerStrip());  // not pixel
     auto const& bounds = stripdet->specificSurface().bounds();
     auto detid = stripdet->geographicalId();
-    auto& p = stripData[detid];
+    auto  pp = stripData.emplace(std::make_pair(detid,StripData()));
+    auto & p = (*pp.first).second;
     p.det = stripdet;
     p.topology = (StripTopology*)(&stripdet->topology());
     p.drift = getDrift(stripdet);
     p.thickness = bounds.thickness();
     p.nstrips = p.topology->nstrips();
   }
+
+  mallocProfiler::dump(std::cout,' ', mallocProfiler::SortBy::none, mallocProfiler::currentThread);
+  mallocProfiler::deactivate(mallocProfiler::currentThread);
+  mallocProfiler::deactivate(mallocProfiler::allThreads);
+  std::cout << "in ClusterShapeHitFilter::fillStripData " << stripData.size() << ' ' << stripData.bucket_count() << ' ' << stripData.max_bucket_count() << ' ' << stripData.max_load_factor() << std::endl;
+  abort();
 }
```

Raw dump.... (just two lines...)

Malloc Profiler

Local Instrumantation

- setenv LD_PRELOAD /data/user/innocent/MallocProfiler/mallocProfilerOFF.so
- cmsRunGlibC reco1.py
  - in ClusterShapeHitFilter::fillStripData 15148 32 1
  - ClusterShapeHitFilter::fillStripData;operator new;malloc; 11 324840 166024 248208
  - ClusterShapeHitFilter::fillStripData;operator new;malloc; 15148 727104 727104 727104
- Add stripData.reserve(dus.size()-offset);
  - ClusterShapeHitFilter;ClusterShapeHitFilter::fillStripData;std::_Hashtable<unsigned int, std::pair<unsigned int const, ClusterShapeHitFilter::StripData>,std::__detail::_Select1st, std::equal_to<unsigned int>, std::hash<unsigned int>, std::__detail::_Mod_range_hashing, std::__detail::_Default_ranged_hash, std::__detail::_Prime_rehash_policy, std::__detail::_Hashtable_traits<false, false, true> >::_M_rehash;operator new;get_stacktrace; 1 121384 121384 121384
  - ClusterShapeHitFilter::fillStripData;operator new;malloc; 15148 727104 727104 727104
- 727104/15148 = 48 = 32+8 + 8 (bucket?)
- 121384/15173 = 8 (pointer to bucket?)

Experimenting with friendly "detailed" dump

```
in ClusterShapeHitFilter::fillStripData 15148 32 40 1
Stat 1 121384 121384 121384 at
#0 malloc :0
#1 operator new(unsigned long) ../../../../gcc_src/libstdc++-v3/libsupc++/new_op.cc:50
#2 std::__detail::_Hashtable_alloc<std::allocator<std::__detail::_Hash_node<std::pair<unsigned int const, ClusterShapeHitFilter::StripData>, false> > >::_M_allocate_buckets(unsig
ned long) /cvmfs/cms.cern.ch/el8_amd64_gcc12/external/gcc/12.3.1-40d504be6370b5a30e3947a6e575ca28/include/c++/12.3.1/bits/new_allocator.h:137
#3 std::_Hashtable<unsigned int, std::pair<unsigned int const, ClusterShapeHitFilter::StripData>, std::allocator<std::pair<unsigned int const, ClusterShapeHitFilter::StripData> >
, std::__detail::_Select1st, std::equal_to<unsigned int>, std::hash<unsigned int>, std::__detail::_Mod_range_hashing, std::__detail::_Default_ranged_hash,
std::__detail::_Prime_rehash_policy, std::__detail::_Hashtable_traits<false, false, true> >::rehash(unsigned long) /cvmfs/cms.cern.ch/el8_amd64_gcc12/external/gcc/12.3.1-40d504be
6370b5a30e3947a6e575ca28/include/c++/12.3.1/bits/hashtable.h:2523
#4 ClusterShapeHitFilter::ClusterShapeHitFilter(TrackerGeometry const*, TrackerTopology const*, MagneticField const*, SiPixelLorentzAngle const*, SiStripLorentzAngle const*, std:
:__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) /data/user/innocent/CMSSW_13_3_0_pre4/src/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc:54
Stat 15148 727104 727104 727104 at
#0 malloc :0
#1 operator new(unsigned long) ../../../../gcc_src/libstdc++-v3/libsupc++/new_op.cc:50
#2 std::__new_allocator<std::__detail::_Hash_node<std::pair<unsigned int const, ClusterShapeHitFilter::StripData>, false> >::allocate(unsigned long, void const*) /cvmfs/cms.cern.
ch/el8_amd64_gcc12/external/gcc/12.3.1-40d504be6370b5a30e3947a6e575ca28/include/c++/12.3.1/bits/new_allocator.h:137
#3 ClusterShapeHitFilter::ClusterShapeHitFilter(TrackerGeometry const*, TrackerTopology const*, MagneticField const*, SiPixelLorentzAngle const*, SiStripLorentzAngle const*, std:
:__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) /data/user/innocent/CMSSW_13_3_0_pre4/src/RecoTracker/PixelLowPtUtilities/src/ClusterShapeHitFilter.cc:54
```

# Current status and future developments

- What is provided:
  - *mallocProfiler.so*:   the instrumentation library to preload
  - *mallocProfilerOFF.so*: as above with all tracing deactivated
  - *dummyMallocProfiler.so*: a dummy version of the API to link unconditionally
  - *tracingThread.so*: threads dumping summary and stacktraces at regular intervals (to preload)

- What could be provided
  - A wrapping script compiling options in a middleman library
  - A filtering hook
  - CMSSW specific instrumentations
  - More global statistics (mem-tot and mem-live histograms for instance)

- What will not be provided
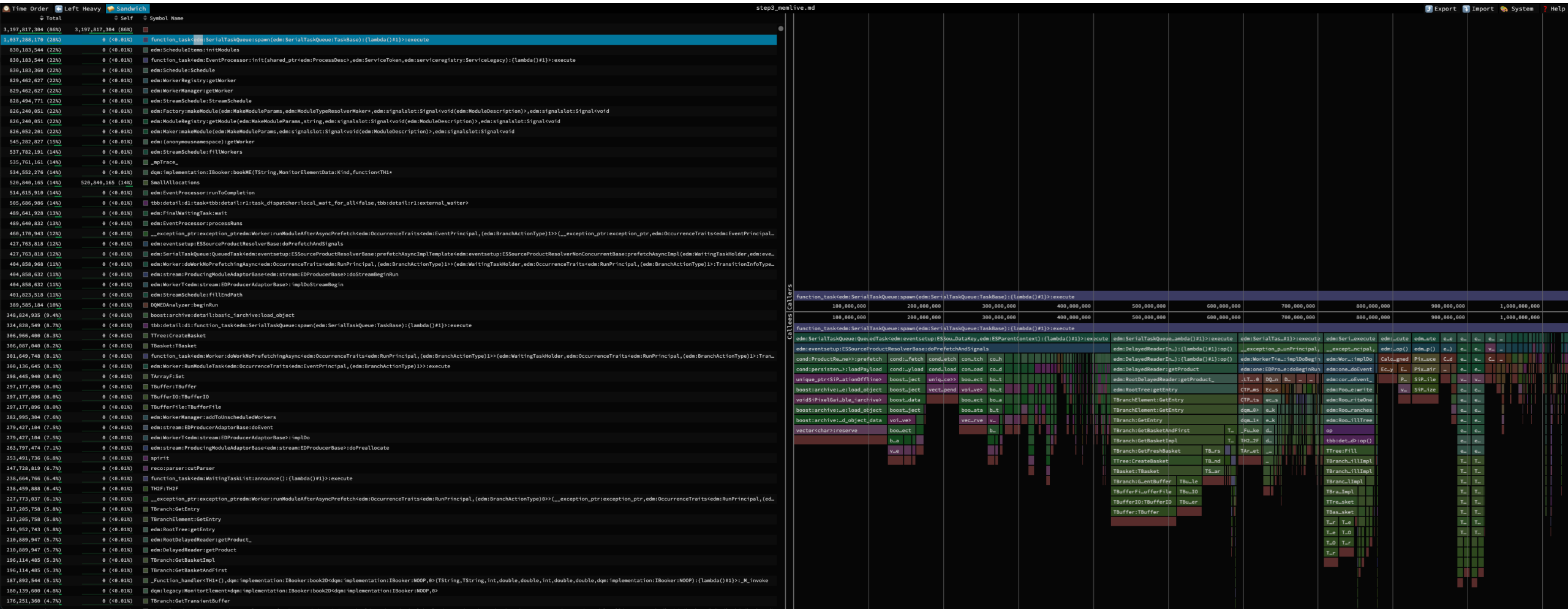  - The possibility to use a configuration file or env-var instead of the API

# BACKUP

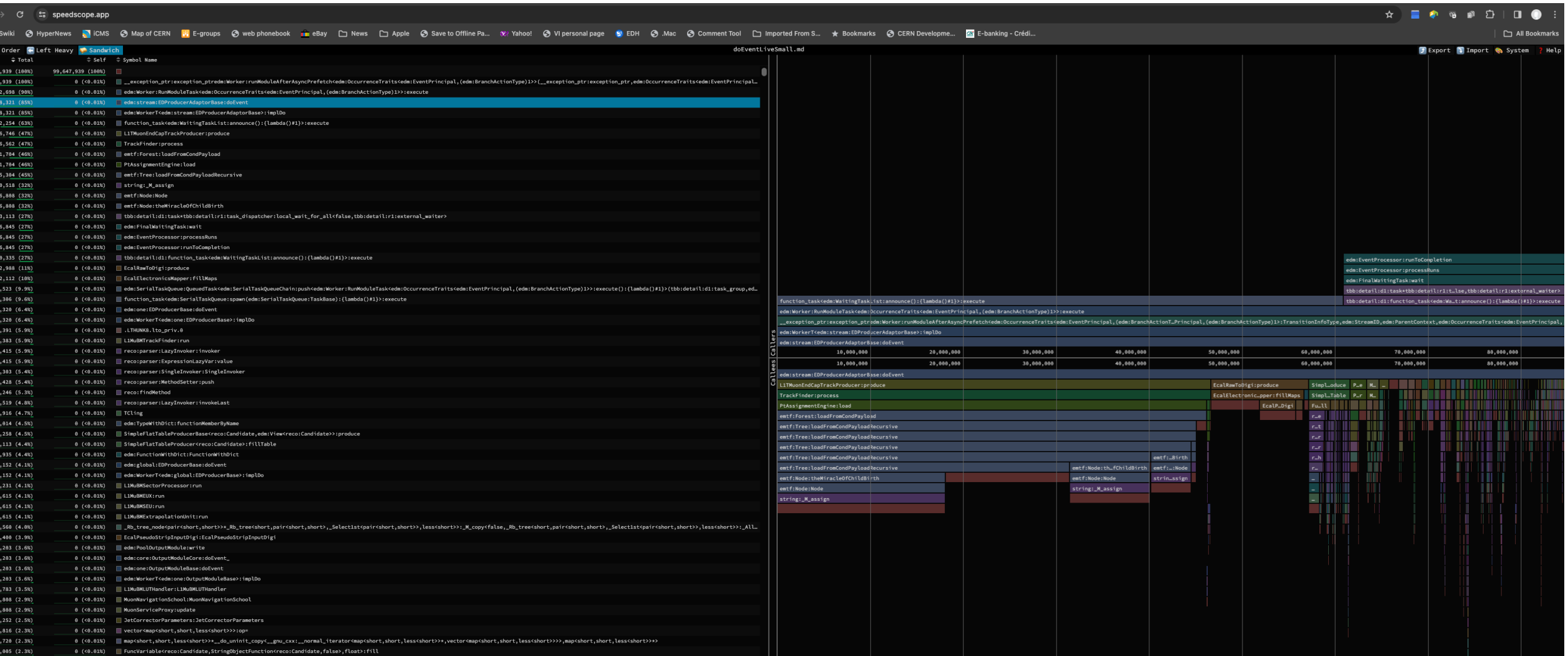# Flamegraph (memtot in "doEvent")
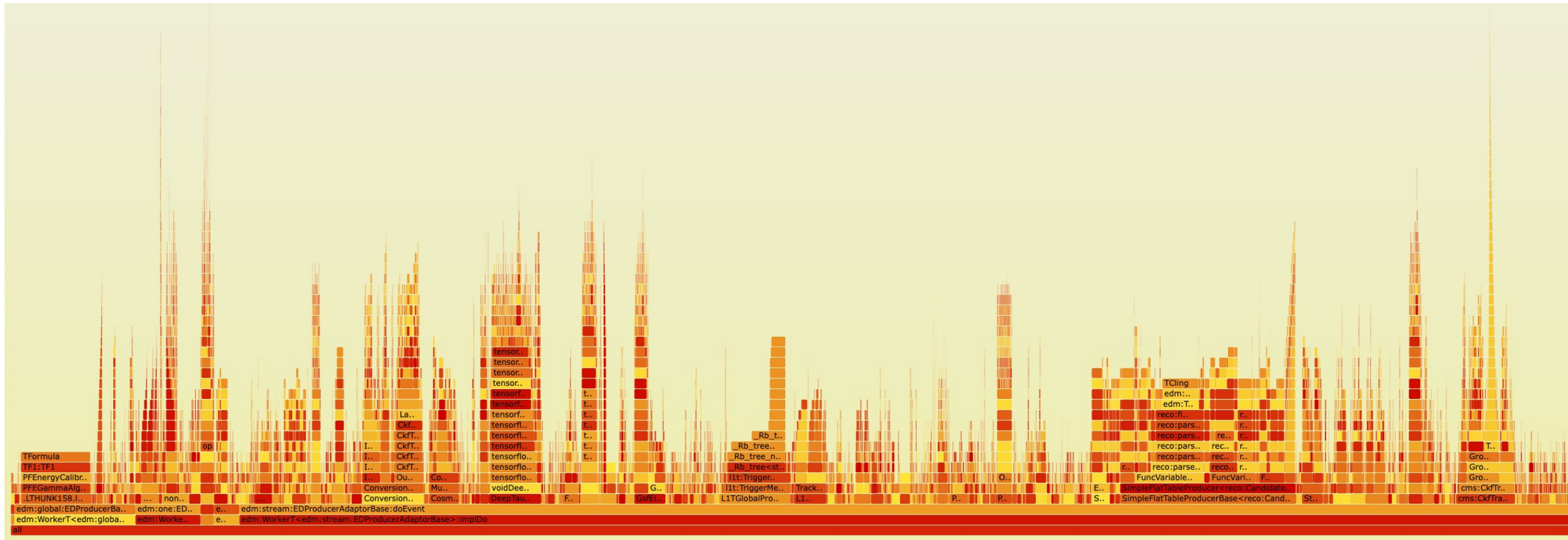
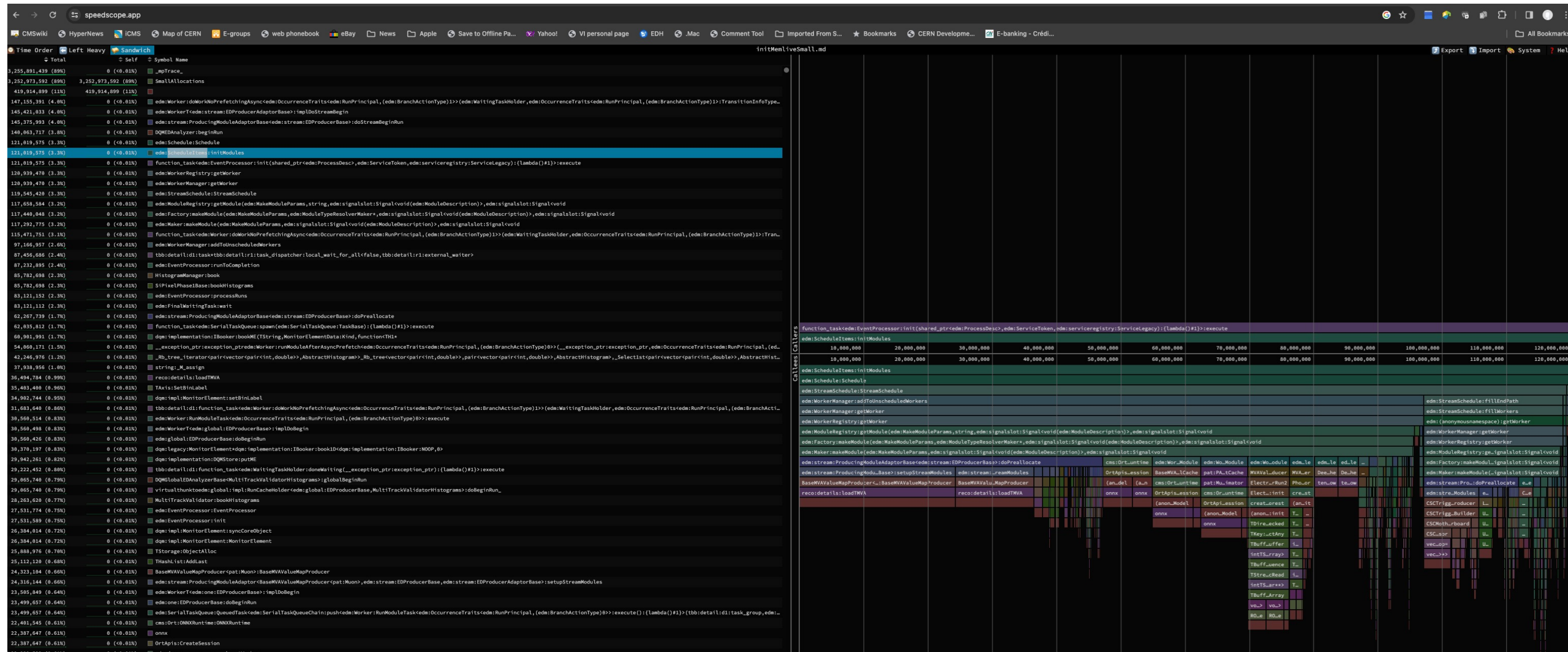# SpeepScope: MemTot >128B in doEvent
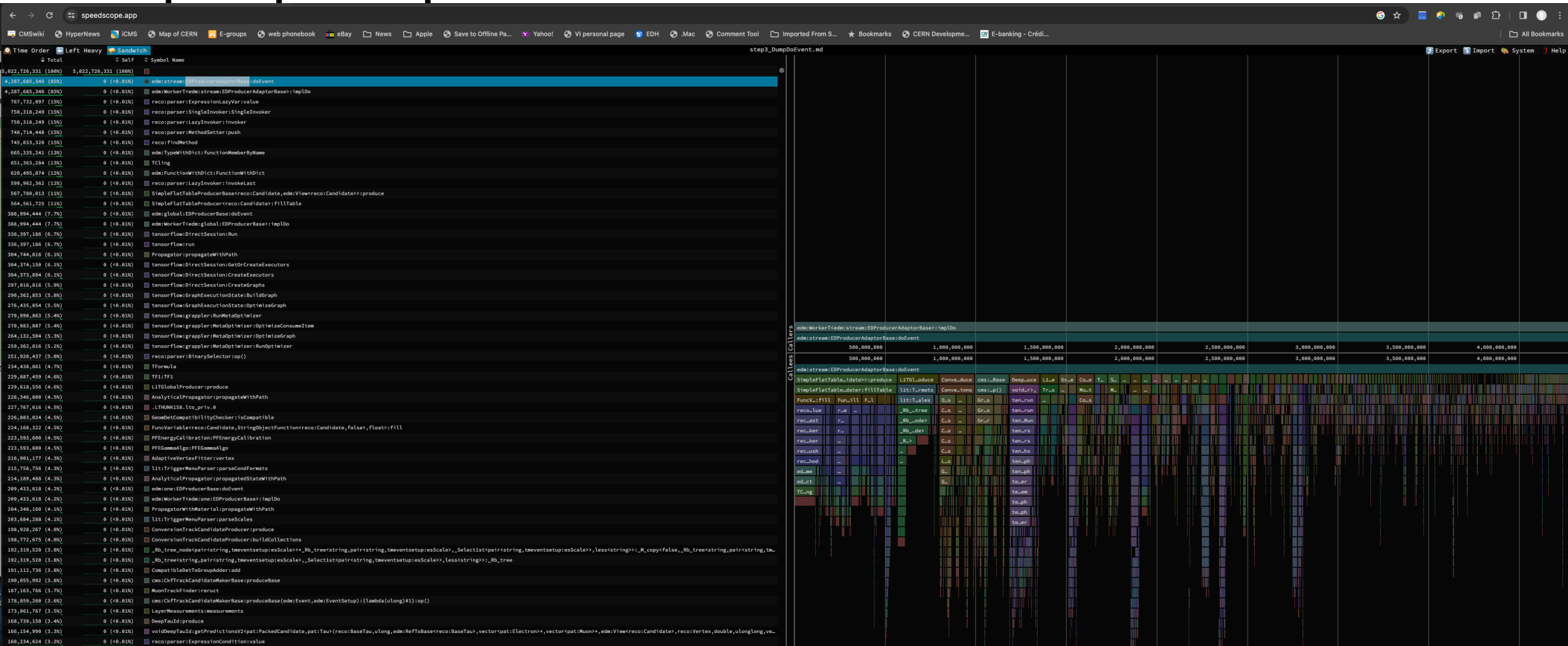
# Speedscope (memlive >128B full)

# Speedscope: memlive SmallAlloc in doEvent

# Speedscope: memlive SmallAlloc NOT in doEvent

grep -v doEvent memliveSmall.md | grep -v spirit | sed 's/onnx.*\;/onnx\;/g' | sed 's/tensorflow.*\;/tensorflow\;/g' | sed 's/TMVA.*\;/TMVA\;/g' | sed 's/Hashtable.*\;/Hashtable\;/g' | grep -v "TClass; "

# tail -n 11 memliveSmall.md

Malloc Profiler