Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Making C++ Run Better with R

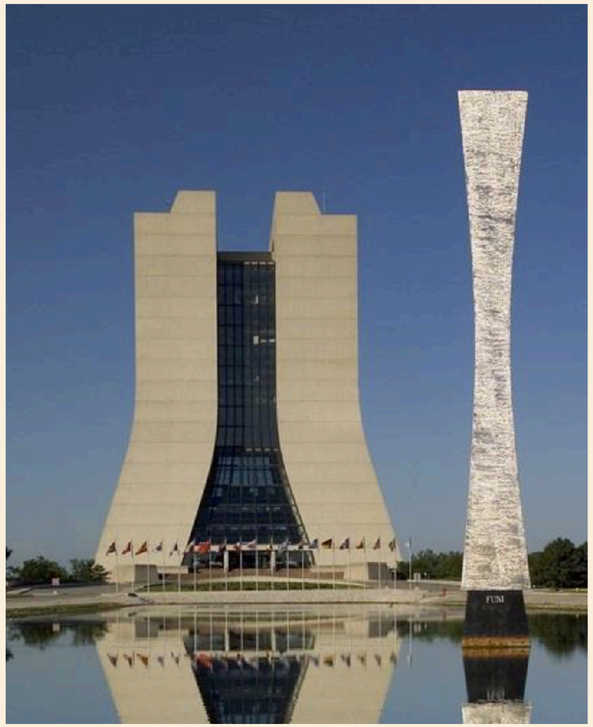
Adam Lyon (Fermilab Scientific Computing Division)

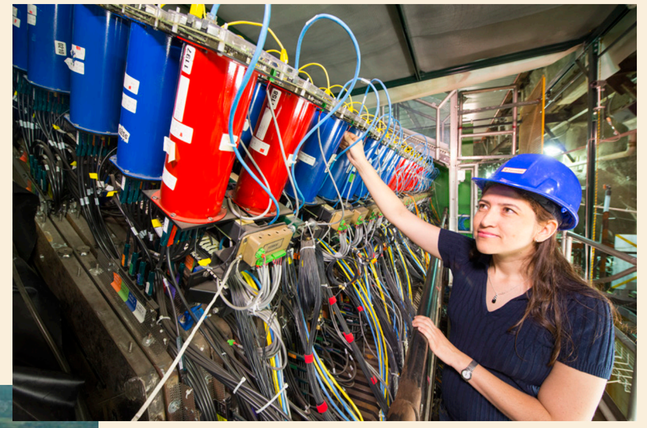
SATRDAYS Chicago

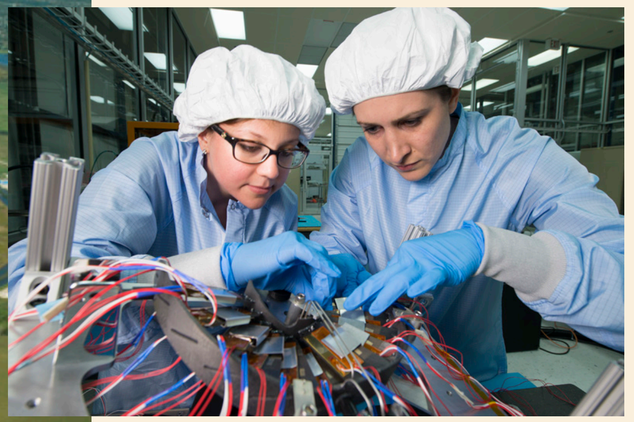
27 April 2019

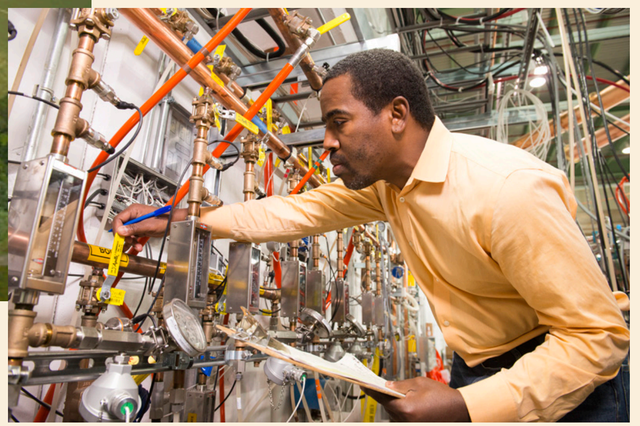
Fermi National Accelerator Laboratory

Fermilab is America's Particle Physics and Accelerator Laboratory





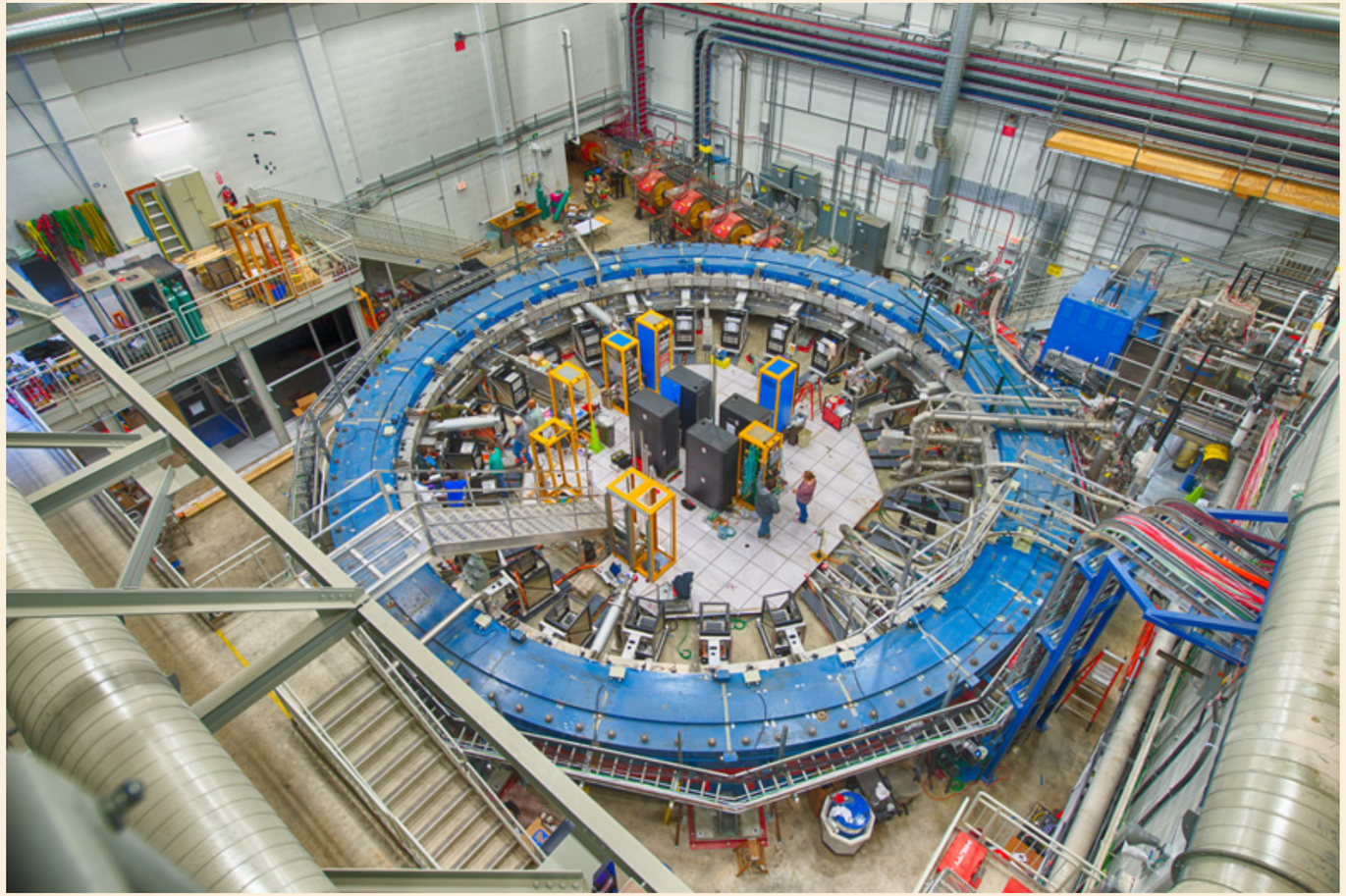


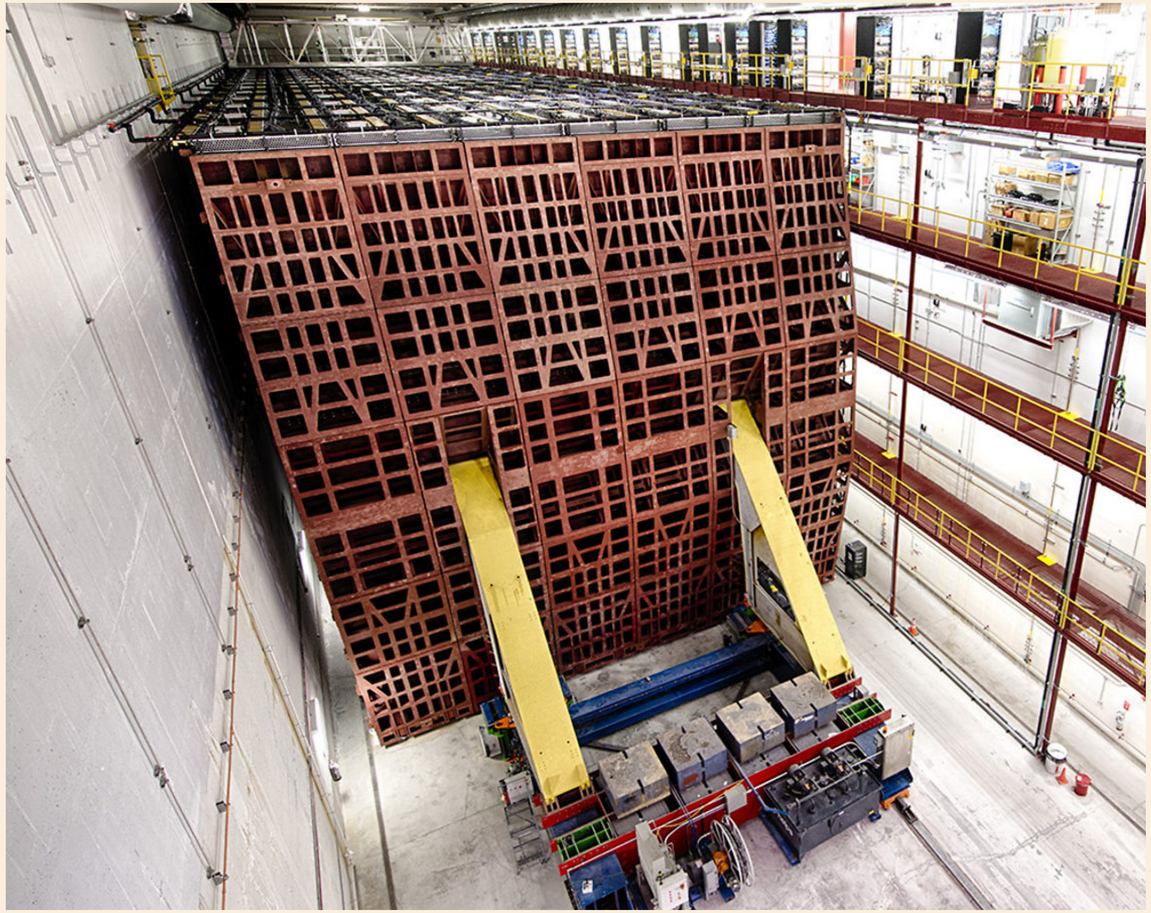


Fermilab

We build huge machines to understand how our universe works

We do science! We use machines to search for smallest things we can find in nature and learn more about them (subatomic particles)





Fermilab

And we write lots of code!

Mostly written by many physicists who *are not* software experts

Except for difficult infrastructure code written by a few computing professionals who *are* software experts

Fermilab

Lots and lots and lots of code…

Millions of lines of C++

Process raw data from experiments into physics objects (reconstruction)

Simulations

Analyze data

All open source

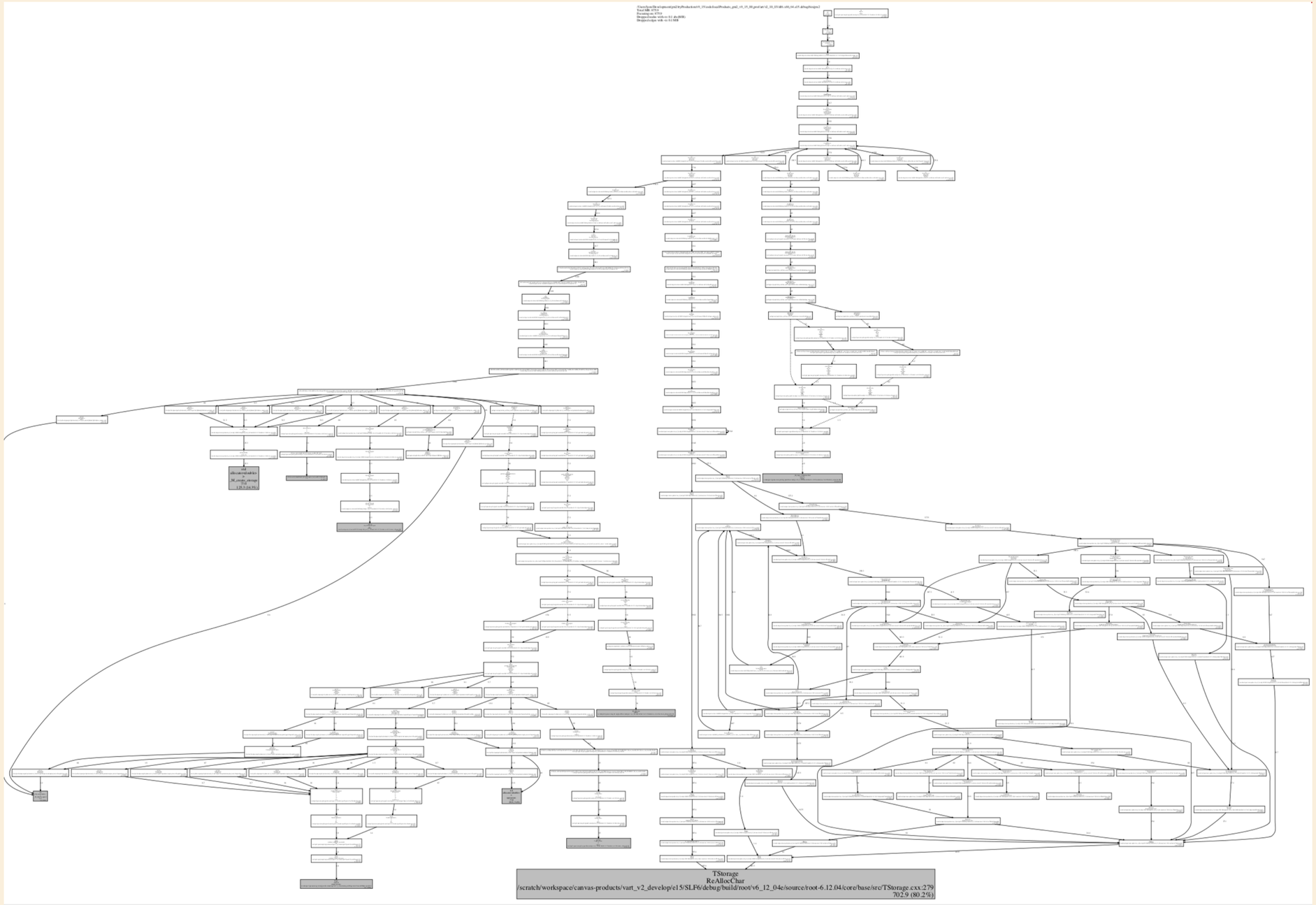
Fermilab

Things can go wrong

Code takes too long to run

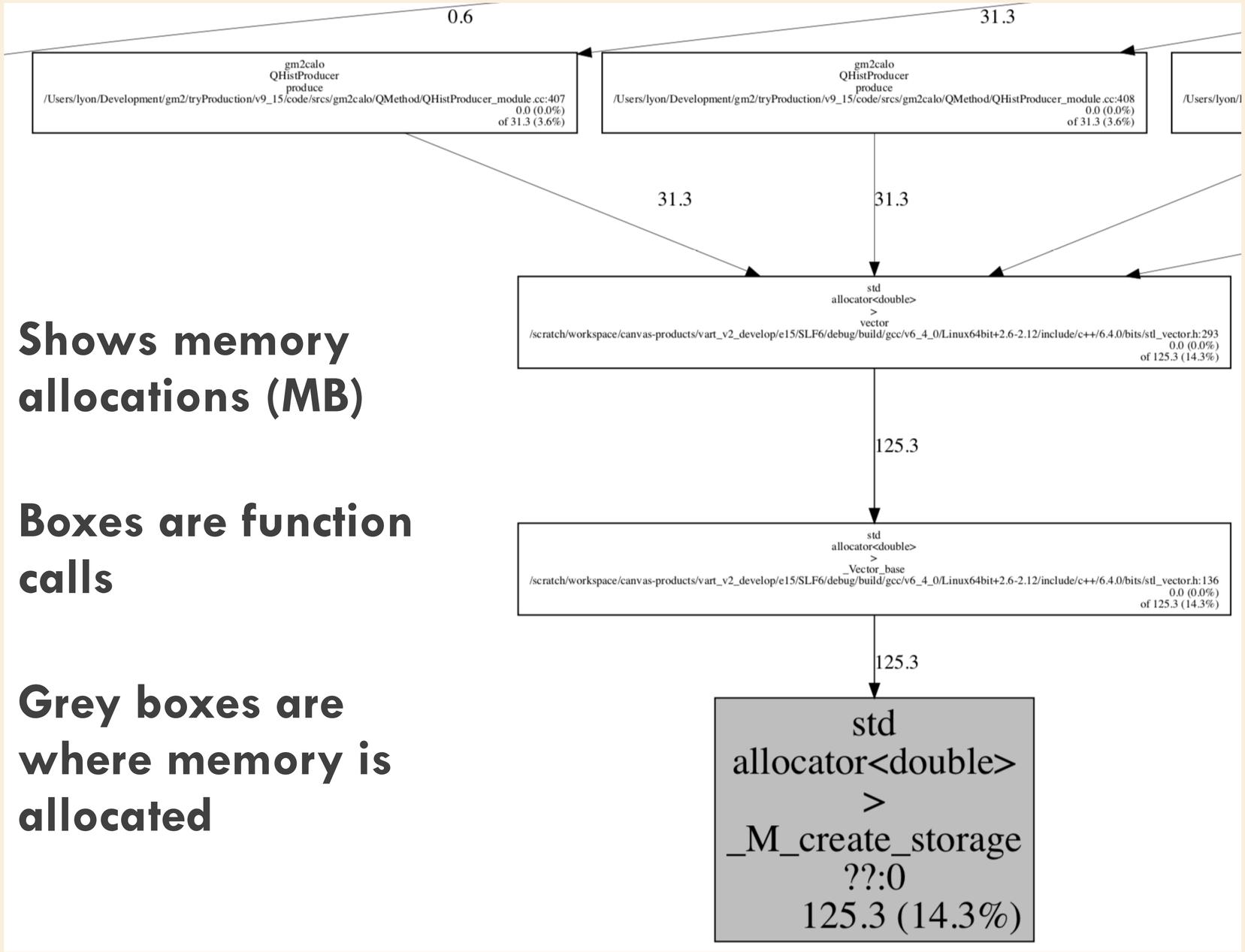
Code uses too much memory

Run a profiler, like Google's `gperftools`, and get this…



Now what?

Fermilab



Shows memory allocations (MB)

Boxes are function calls

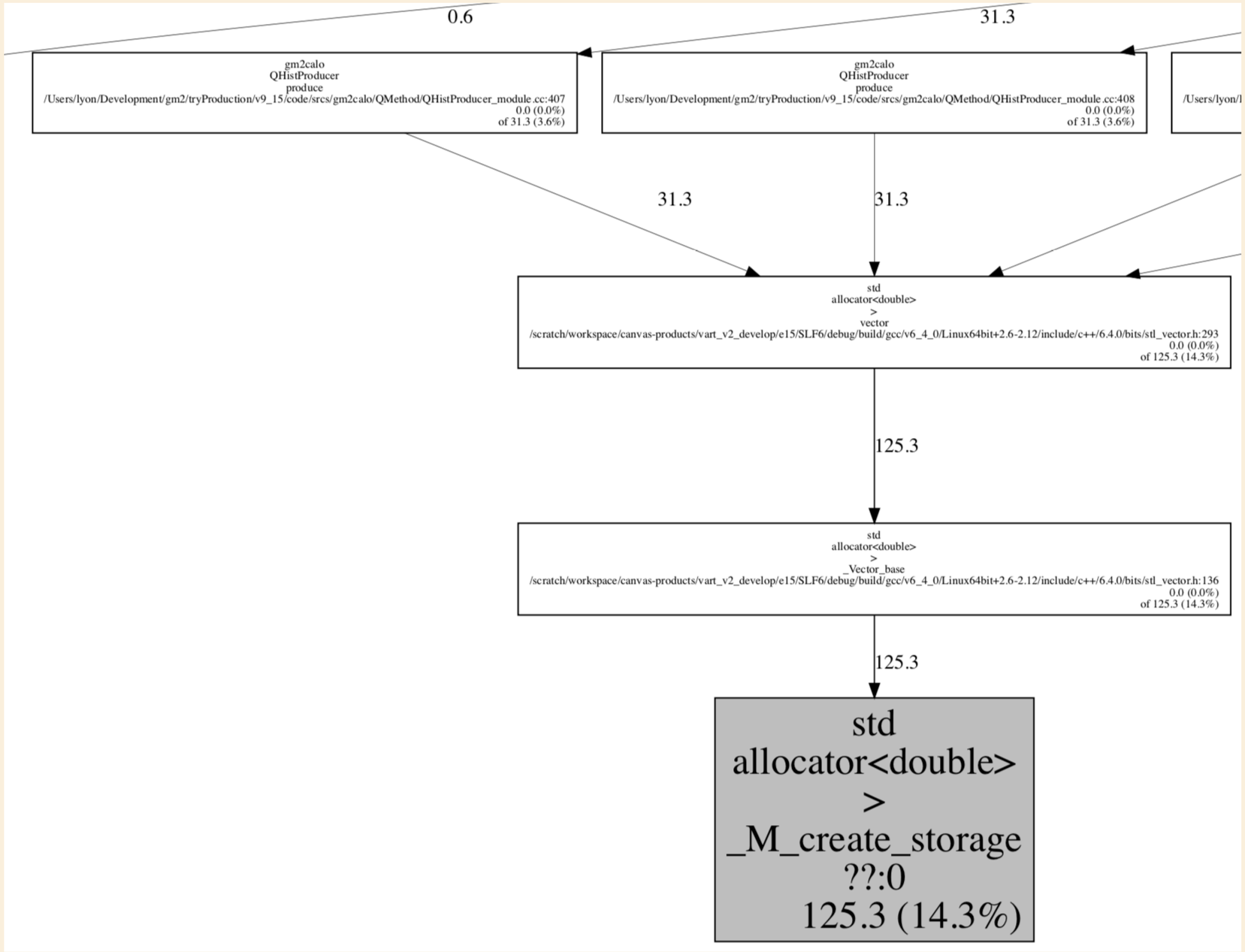
Grey boxes are where memory is allocated

Zooming in helps, a little

How to analyze programmatically?

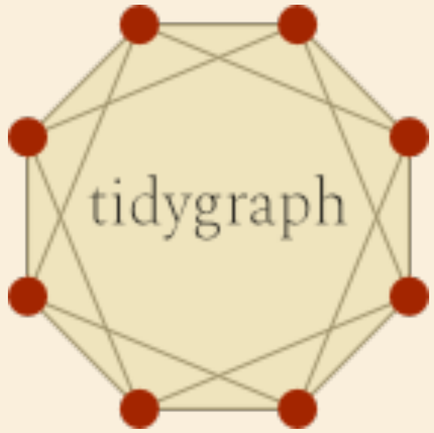
How to analyze reproducibly?

Fermilab



This looks like a directed graph

R can do that!



Fermilab

Analyzing gperftools output with Tidygraph

gperftools output → Tidygraph

1. `$ dot -Tdot_json out.dot > out.dot.json # Convert dot file into json`
2. `jsonlite::fromJSON(out.dot.json)  # Read into R`
3. *Extract nodes (boxes) and edges (lines) to data frames and "fix them up"*
4. `g ← tidygraph::tbl_graph(nodes=nodes, edges=edges)  # Make Tidygraph`

Do stuff: Find the memory allocators (bottom of the graphs)

```
g %>%
  filter(node_is_sink()) %>%
  as_tibble() %>%
  select(index, name, totalMem, totalPerc) %>%
  arrange(-totalMem)
```

index <int>	name <chr>	totalMem <dbl>	totalPerc <dbl>
1	TStorage ReAllocChar	702.9	80.2
71	std allocator<double> > _M_create_storage	125.3	14.3
109	std allocator<double> > operator=	29.9	3.4
148	_M_range_initialize<char const*> (inline)	4.9	0.6
152	Eigen internal aligned_malloc	4.4	0.5
155	_M_allocate (inline)	3.8	0.4
169	std allocator<double> > _M_default_append	3.4	0.4
200	gm2calo TemplateFitter> > > allocate	0.5	0.1
202	TBufferFile WriteFastArray@176fe6	0.5	0.1
205	TBufferFile WriteFastArray@176c32	0.3	0.0

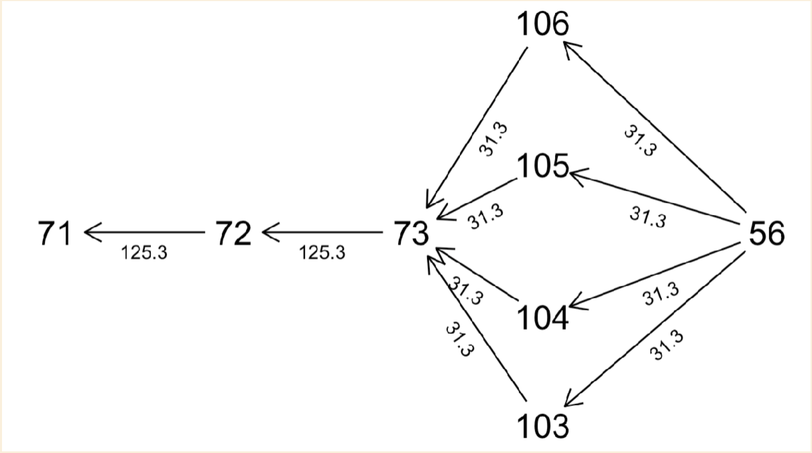
Traverse up from one of them

```
g %>%
  mutate(dist = dfs_dist(71, mode="in")) %>%
  filter(!is.na(dist)) %>%
  as_tibble() %>%
  arrange(dist) %>%
  select(index, name, file, totalMem, dist)
```

This is readable, programmable, reproducible

index	name	file	totalMem	dist
71	std allocator<double> > _M_create_storage	??:0	125.3	0
72	std allocator<double> > _Vector_base	stl_vector.h:136	125.3	1
73	std allocator<double> > vector	stl_vector.h:293	125.3	2
103	gm2calo QHistProducer produce	QHistProducer_module.cc:406	31.3	3
104	gm2calo QHistProducer produce	QHistProducer_module.cc:407	31.3	3
105	gm2calo QHistProducer produce	QHistProducer_module.cc:408	31.3	3
106	gm2calo QHistProducer produce	QHistProducer_module.cc:409	31.3	3
56	_ZN3art10EDProducer7doEventERNS_14EventPrincipal…	EDProducer.cc:25	168.1	4

Traverse up from one of them



Make easier to read pictures too

index	name	file	totalMem	dist
71	std allocator<double> > _M_create_storage	??:0	125.3	0
72	std allocator<double> > _Vector_base	stl_vector.h:136	125.3	1
73	std allocator<double> > vector	stl_vector.h:293	125.3	2
103	gm2calo QHistProducer produce	QHistProducer_module.cc:406	31.3	3
104	gm2calo QHistProducer produce	QHistProducer_module.cc:407	31.3	3
105	gm2calo QHistProducer produce	QHistProducer_module.cc:408	31.3	3
106	gm2calo QHistProducer produce	QHistProducer_module.cc:409	31.3	3
56	_ZN3art10EDProducer7doEventERNS_14EventPrincipal…	EDProducer.cc:25	168.1	4

Fermilab

Here are the culprits (allocates ~15% of total memory usage)

```
405             xtalN_[iCalo].push_back(iChan);
406             e_Traces_[iCalo].push_back(std::vector<double>(nBins_[iCalo], 0));
407             eSqSum_Traces_[iCalo].push_back(std::vector<double>(nBins_[iCalo], 0));
408             rawSum_Traces_[iCalo].push_back(std::vector<double>(nBins_[iCalo], 0));
409             pedSum_Traces_[iCalo].push_back(std::vector<double>(nBins_[iCalo], 0));
```

Takeaways:

Make interactive tasks programmable and reproducible

Easy to document. Easy to share. Easy for other people to try and extend.

See http://bit.ly/gperftoolsR

Fermilab is more than physics — open to the public — come visit!

Fermilab