Analyzing compile-time overhead

Hadrien Grasland

2022-07-05

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004761.

First, a little story

- In ACTS, we tried to follow **Modern C++ Best Practices™**
  - Avoid vtables like pest, use templates all the way down
  - Put most code in headers because templates + inlining
  - Use fancy metaprograms to reduce runtime work

First, a little story

- In ACTS, we tried to follow **Modern C++ Best Practices™**
  - Avoid vtables like pest, use templates all the way down
  - Put most code in headers because templates + inlining
  - Use fancy metaprograms to reduce runtime work
- It worked ok for years, then **weird CI crashes** creeped in
  - Job abruptly fails, not always in the same place, no error log
  - Reducing concurrency for reproductibility, it goes away
  - …until someday it doesn't and the crashes are back at -j1

Investigation begins

- « Luckily », several developers got **similar local crashes**
  - Easier to figure out they’re running out of RAM
  - Often decide to live with slower builds by tuning down -j
  - Friendly colleague suggests downloading more RAM

Investigation begins

- « Luckily », several developers got **similar local crashes**
  - Easier to figure out they're running out of RAM
  - Often decide to live with slower builds by tuning down -j
  - Friendly colleague suggests downloading more RAM
- …but none of these workarounds will work for Github CI VMs !
  - Need to admit that **our codebase is the problem**
  - Problem is, there are about 500k lines of code in it
  - How to efficiently figure out which of these are the culprits ?

Build profiling howto

Step 1 : Set a goal

- Minimal sanity : **Don’t crash CI**
  - If you use Github CI, you have 2 cores & 7 GB
  - So even -j1 build must use <6 GB (keeping 1GB for OS)
  - If you want -j2 builds, get below 3 GB/process

Step 1 : Set a goal

- Minimal sanity : **Don’t crash CI**
  - If you use Github CI, you have 2 cores & 7 GB
  - So even -j1 build must use <6 GB (keeping 1GB for OS)
  - If you want -j2 builds, get below 3 GB/process
- Optimal sanity : **Use dev machines efficiently**
  - Classic mid-end laptop has 2 GB RAM per (logical) core
  - So for 4 threads / 8 GB, try to get below 1,5 GB/process*

* The typical dev computer has more background tasks → 2GB for OS+background is safer

Step 2 : Pick pathological files

- There is no reliable* tool to analyze your full build
  - Best to focus on individual compilation units (cpp files)
- How to find bloated compilation units ?
  - Easy but tedious : Watch system monitor during build
  - Automated : Use **cmakeperf** (Paul's part)

* https://github.com/aras-p/ClangBuildAnalyzer is nice but does not scale…

Step 3 : Find a proxy for RAM usage

- Compilers won't tell you anything about RAM consumption
  - You don't want to run e.g. heaptrack on those beasts*
- However, **what consumes RAM, most likely takes time**
  - And we do have time profiles, via clang's -ftime-trace
  - Well, not quite a time profile, but an activity history
  - However, the information we need is hidden in there

* Learning that memory is consumed while instantiating *some* templated code is not useful

Step 4 : Make the profile readable

- Remember, we're in this situation due to excessive templating
- So the profile will contain type/function names like this :

- Useful info (e.g. method name) often some way through
  - Need an AST, and thus a parser, to reliably isolate it…

Step 5 : Profit !

```
Hierarchical profile:
ExecuteCompiler [2:30.12, 100.00%]
├─Backend [1:20.95, 53.93%]
│ ├─PerModulePasses [56.05s, 37.34%]
│ │ └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFinder/CombinatorialKalmanFilterTests.cpp) [56.05s, 37.34%]
│ │   └─…13301 callee(s) below 3.00%…
│ ├─CodeGenPasses [22.96s, 15.30%]
│ │ └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFinder/CombinatorialKalmanFilterTests.cpp) [22.90s, 15.25%]
│ │   └─…2522 callee(s) below 3.00%…
│ └─…1 callee(s) below 3.00%…
├─Frontend [43.35s, 28.88%]
│ ├─PerformPendingInstantiations [31.99s, 21.31%]
│ │ ├─InstantiateFunction(…::findTracks<…>) [15.65s, 10.43%]
│ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [15.37s, 10.24%]
│ │ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate_impl<…::PropagatorResult<…>, …::State<…>>) [15.00s, 10.00%]
│ │ │ │ │ ├─InstantiateFunction(Acts::ActionList<…>::operator()<…::State<…>, …::EigenStepper<…>, …::PropagatorResult<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ ├─InstantiateFunction(Acts::detail::action_list_impl<…>::action<…::tuple<…>, …::PropagatorResult<…>, …::State<…>, …::EigenStepper<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ │ └─InstantiateFunction(…::action<…>) [14.68s, 9.78%]
│ │ │ │ │ │ │   └─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ │     ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<…>>) [13.01s, 8.66%]
│ │ │ │ │ │ │     │ ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::addSourcelinkState) [7.58s, 5.05%]
│ │ │ │ │ │ │     │ │ ├─InstantiateFunction(Acts::GainMatrixUpdater<…>::operator()<…::TrackStateProxy<…>>) [6.32s, 4.21%]
│ │ │ │ │ │ │     │ │ │ ├─InstantiateFunction(…::template_switch<…>) [5.46s, 3.63%]
│ │ │ │ │ │ │     │ │ │ │ ├─InstantiateFunction(…::template_switch<…>) [4.56s, 3.04%]
│ │ │ │ │ │ │     │ │ │ │ │ └─…35 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ │ │ └─…34 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ │ └─…50 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ └─…77 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ └─InstantiateFunction(Acts::CKFSourceLinkSelector::operator()<…::VoidMeasurementCalibrator, …::ExtendedMinimalSourceLink>) [5.30s, 3.53%]
│ │ │ │ │ │ │     │   └─…177 callee(s) below 3.00%…
│ │ │ │ │ │ │     └─…23 callee(s) below 3.00%…
│ │ │ │ │ │ └─…23 callee(s) below 3.00%…
│ │ │ │ │ └─…1 callee(s) below 3.00%…
│ │ │ │ └─…21 callee(s) below 3.00%…
│ │ │ └─…12 callee(s) below 3.00%…
│ │ └─…32 callee(s) below 3.00%…
│ └─…425 callee(s) below 3.00%…
└─…58 callee(s) below 3.00%…
└─Frontend [25.74s, 17.15%]
  └─…9599 callee(s) below 3.00%…
```

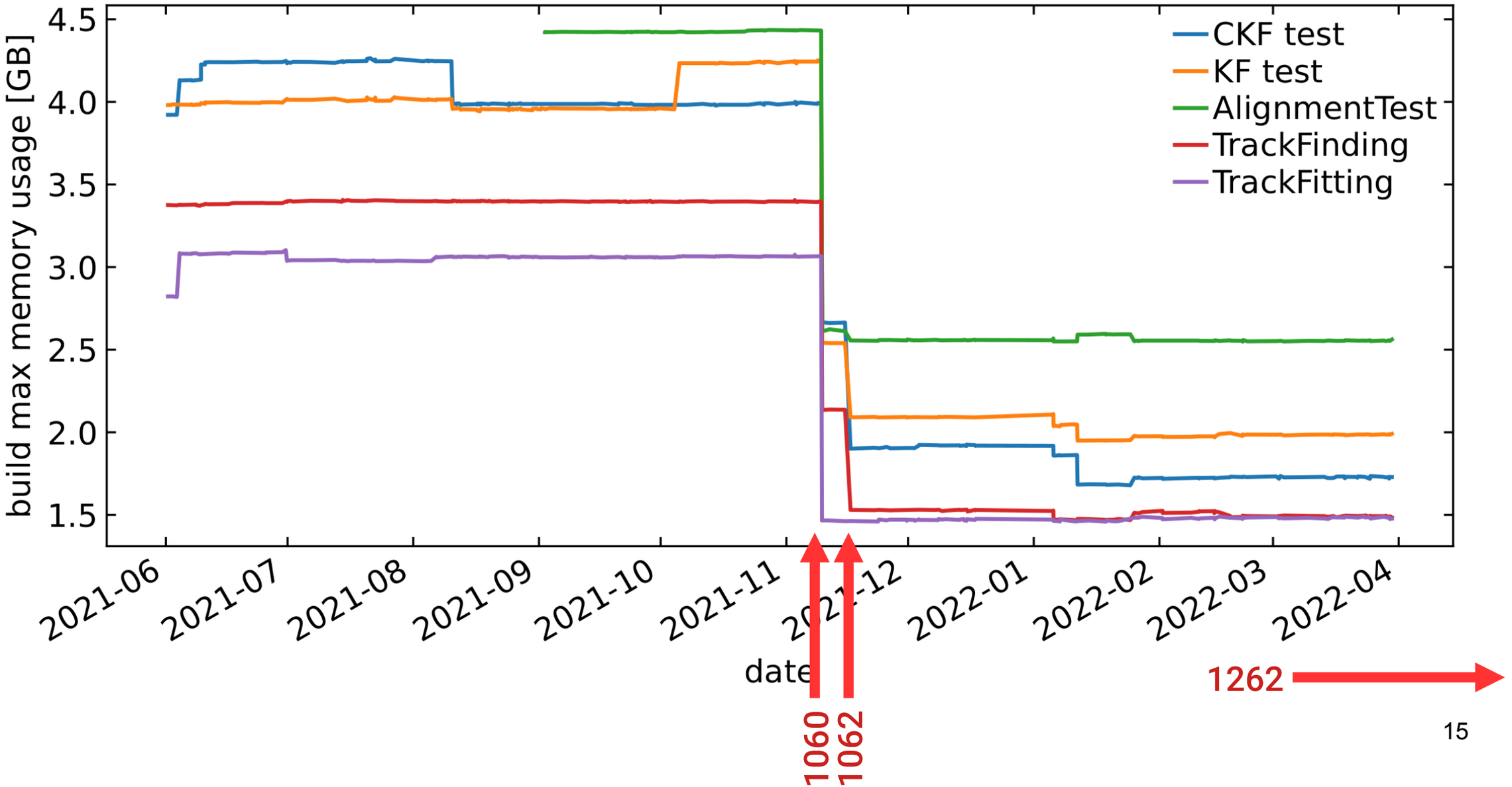
Step 5 : Profit !

```
Hierarchical profile:
ExecuteCompiler [2:30.12, 100.00%]
├─Backend [1:20.95, 53.93%]
│ ├─PerModulePasses [56.05s, 37.34%]
│ │ └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFinder/CombinatorialKalmanFilterTests.cpp) [56.05s, 37.34%]
│ │   └─…13301 callee(s) below 3.00%…
│ ├─CodeGenPasses [22.96s, 15.30%]
│ │ └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFinder/CombinatorialKalmanFilterTests.cpp) [22.90s, 15.25%]
│ │   └─…2522 callee(s) below 3.00%…
│ └─…1 callee(s) below 3.00%…
├─Frontend [43.35s, 28.88%]
│ ├─PerformPendingInstantiations [31.99s, 21.31%]
│ │ ├─InstantiateFunction(…::findTracks<…>) [15.65s, 10.43%]
│ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [15.37s, 10.24%]
│ │ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate_impl<…::PropagatorResult<…>, …::State<…>>) [15.00s, 10.00%]
│ │ │ │ │ ├─InstantiateFunction(Acts::ActionList<…>::operator()<…::State<…>, …::EigenStepper<…>, …::PropagatorResult<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ ├─InstantiateFunction(Acts::detail::action_list_impl<…>::action<…::tuple<…>, …::PropagatorResult<…>, …::State<…>, …::EigenStepper<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ │ └─InstantiateFunction(…::action<…>) [14.68s, 9.78%]
│ │ │ │ │ │ │   └─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<…>>) [14.68s, 9.78%]
│ │ │ │ │ │ │     ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<…>>) [13.01s, 8.66%]
│ │ │ │ │ │ │     │ ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::addSourcelinkState) [7.58s, 5.05%]
│ │ │ │ │ │ │     │ │ ├─InstantiateFunction(Acts::GainMatrixUpdater<…>::operator()<…::TrackStateProxy<…>>) [6.32s, 4.21%]
│ │ │ │ │ │ │     │ │ │ ├─InstantiateFunction(…::template_switch<…>) [5.46s, 3.63%]
│ │ │ │ │ │ │     │ │ │ │ ├─InstantiateFunction(…::template_switch<…>) [4.56s, 3.04%]
│ │ │ │ │ │ │     │ │ │ │ │ └─…35 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ │ │ └─…34 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ │ └─…50 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ │ └─…77 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ ├─InstantiateFunction(Acts::CKFSourceLinkSelector::operator()<…::VoidMeasurementCalibrator, …::ExtendedMinimalSourceLink>) [5.30s, 3.53%]
│ │ │ │ │ │ │     │ │ └─…177 callee(s) below 3.00%…
│ │ │ │ │ │ │     │ └─…23 callee(s) below 3.00%…
│ │ │ │ │ │ │     └─…23 callee(s) below 3.00%…
│ │ │ │ │ │ └─…1 callee(s) below 3.00%…
│ │ │ │ │ └─…21 callee(s) below 3.00%…
│ │ │ │ └─…12 callee(s) below 3.00%…
│ │ │ └─…32 callee(s) below 3.00%…
│ │ └─…425 callee(s) below 3.00%…
│ └─…58 callee(s) below 3.00%…
└─Frontend [25.74s, 17.15%]
  └─…9599 callee(s) below 3.00%…
```

...and this also has a WIP perf report style interactive viz :)

Application to ACTS

Let’s look at some PRs



Before Acts#1060 (SourceLink rework)

```
Frontend [14.50s, 28.33%]
  PerformPendingInstantiations [11.10s, 21.70%]
    InstantiateFunction(…::findTracks<…>) [8.28s, 16.19%]
      InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [8.15s, 15.93%]
        InstantiateFunction(Acts::Propagator<…>::propagate_impl<…::PropagatorResult<…>, …::State<…>>) [7.91s, 15.46%]
          InstantiateFunction(Acts::ActionList<…>::operator()<…::State<…>, …::EigenStepper<>, …::PropagatorResult<…>>) [7.85s, 15.34%]
            InstantiateFunction(Acts::detail::action_list_impl<…>::action<…::tuple<…>, …::PropagatorResult<…>, …::State<…>, …::EigenStepper<>>) [7.85s, 15.34%]
              InstantiateFunction(…::action<…>) [7.85s, 15.34%]
                InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<>>) [7.85s, 15.34%]
                  InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<>>) [7.06s, 13.79%]
                    InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::addSourcelinkState) [4.36s, 8.51%]
                      InstantiateFunction(Acts::GainMatrixUpdater::operator()<…::TestSourceLink, 6>) [3.95s, 7.71%]
                        InstantiateFunction(…::visit_measurement<…>) [1.28s, 2.50%]
                          InstantiateFunction(…::template_switch<…>) [1.28s, 2.50%]
                            InstantiateFunction(…::template_switch<…>) [1.06s, 2.07%]
                              InstantiateFunction(…::template_switch<…>) [831.97ms, 1.63%]
                                …2 callee(s) below 1.54%…
                              …1 callee(s) below 1.54%…
                            …1 callee(s) below 1.54%…
                        …158 callee(s) below 1.54%…
                      …16 callee(s) below 1.54%…
                    InstantiateFunction(Acts::MeasurementSelector::operator()<…::variant<…>>) [2.57s, 5.02%]
                      InstantiateFunction(Eigen::DenseBase<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>::eval) [824.60ms, 1.61%]
                        InstantiateFunction(Eigen::Matrix<double, 1, 1, 0>::Matrix<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>) [824.42ms, 1.61%]
                          InstantiateFunction(Eigen::PlainObjectBase<…>::_init1<…::Product<…>, …::Product<…>>) [823.15ms, 1.61%]
                            InstantiateFunction(Eigen::PlainObjectBase<…::Matrix<…>>::_set_noalias<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>) [823.08ms, 1.61%]
                              InstantiateFunction(Eigen::internal::call_assignment_no_alias<…::Matrix<…>, …::Product<…>, …::assign_op<…>>) [822.96ms, 1.61%]
                                InstantiateFunction(Eigen::internal::Assignment<…::Matrix<…>, …::Product<…>, …::assign_op<…>, …::Dense2Dense>::run) [822.48ms, 1.61%]
                                  InstantiateFunction(Eigen::internal::generic_product_impl<…>::evalTo<…::Matrix<…>>) [822.04ms, 1.61%]
                                    InstantiateFunction(Eigen::DenseBase<…::CwiseBinaryOp<…>>::sum) [806.22ms, 1.58%]
                                      InstantiateFunction(Eigen::DenseBase<…::CwiseBinaryOp<…>>::redux<Eigen::internal::scalar_sum_op<double>>) [805.23ms, 1.57%]
                                        InstantiateFunction(Eigen::internal::redux_evaluator<…::CwiseBinaryOp<…>>::redux_evaluator) [790.00ms, 1.54%]
                                          InstantiateFunction(Eigen::internal::evaluator<…::CwiseBinaryOp<…>>::evaluator) [789.93ms, 1.54%]
                                            InstantiateFunction(Eigen::internal::binary_evaluator<…>::binary_evaluator) [789.87ms, 1.54%]
                                              InstantiateFunction(Eigen::internal::binary_evaluator<…>::Data::Data) [789.50ms, 1.54%]
                                                InstantiateFunction(Eigen::internal::evaluator<const Eigen::Transpose<const …::Product<…>>>::evaluator) [789.25ms, 1.54%]
                                                  InstantiateFunction(Eigen::internal::evaluator<Eigen::Transpose<const …::Product<…>>>::evaluator) [789.20ms, 1.54%]
                                                    InstantiateFunction(Eigen::internal::unary_evaluator<…>::unary_evaluator) [789.15ms, 1.54%]
                                                      InstantiateFunction(Eigen::internal::evaluator<const …::Product<…>>::evaluator) [789.01ms, 1.54%]
                                                        InstantiateFunction(Eigen::internal::evaluator<…::Product<…>>::evaluator) [788.96ms, 1.54%]
                                                          InstantiateFunction(Eigen::internal::product_evaluator<…>::product_evaluator) [788.91ms, 1.54%]
                                                            …3 callee(s) below 1.54%…
```

After Acts#1060 (SourceLink rework)

```
├─Frontend [8.79s, 30.80%]
│ ├─PerformPendingInstantiations [5.59s, 19.60%]
│ │ ├─InstantiateFunction(…::findTracks<…>) [3.07s, 10.77%]
│ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [2.97s, 10.40%]
│ │ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate_impl<…::PropagatorResult<…>, …::State<…>>) [2.77s, 9.70%]
│ │ │ │ │ ├─InstantiateFunction(Acts::ActionList<…>::operator()<…::State<…>, …::EigenStepper<>, …::PropagatorResult<…>>) [2.72s, 9.51%]
│ │ │ │ │ │ └─InstantiateFunction(Acts::detail::action_list_impl<…>::action<…::tuple<…>, …::PropagatorResult<…>, …::State<…>, …::EigenStepper<>>) [2.71s, 9.51%]
│ │ │ │ │ │   └─InstantiateFunction(…::action<…>) [2.71s, 9.51%]
│ │ │ │ │ │     └─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<>>) [2.71s, 9.51%]
│ │ │ │ │ │       ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<>>) [2.67s, 9.37%]
│ │ │ │ │ │       │ ├─InstantiateFunction(Acts::MeasurementSelector::operator()<…::variant<…>>) [2.24s, 7.83%]
│ │ │ │ │ │       │ │ └─…41 callee(s) below 2.50%…
```

- Using the same temporal display cutoff as before (~800ms)
  - Frontend time was reduced by 5,71s (-39 %)
  - Max-RSS was reduced by 1,3 GB (-32 %)
- Why ?
  - Mostly loss of CKF::Actor::addSourceLinkState (-30%)
  - As Paul will explain, it could be moved to a dedicated CU

Before Acts#1062 (Delegate @ CKF)

```
Frontend [9.41s, 29.81%]
PerformPendingInstantiations [6.24s, 19.75%]
InstantiateFunction(…::findTracks<…>) [3.68s, 11.66%]
InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [3.57s, 11.31%]
InstantiateFunction(Acts::Propagator<…>::propagate_impl<…::PropagatorResult<…>, …::State<…>>) [3.34s, 10.57%]
InstantiateFunction(Acts::ActionList<…>::operator()<…::State<…>, …::EigenStepper<>, …::PropagatorResult<…>>) [3.27s, 10.37%]
InstantiateFunction(Acts::detail::action_list_impl<…>::action<…::tuple<…>, …::PropagatorResult<…>, …::State<…>, …::EigenStepper<>>) [3.27s, 10.36%]
InstantiateFunction(…::action<…>) [3.27s, 10.36%]
InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<>>) [3.27s, 10.36%]
InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<>>) [3.23s, 10.22%]
InstantiateFunction(Acts::MeasurementSelector::operator()<…::variant<…>>) [2.70s, 8.55%]
InstantiateFunction(Eigen::DenseBase<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>::eval) [881.10ms, 2.79%]
InstantiateFunction(Eigen::Matrix<double, 1, 1, 0>::Matrix<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>) [880.92ms, 2.79%]
InstantiateFunction(Eigen::PlainObjectBase<…>::_init1<…::Product<…>, …::Product<…>>) [879.58ms, 2.79%]
InstantiateFunction(Eigen::PlainObjectBase<…::Matrix<…>>::_set_noalias<Eigen::Product<…::Product<…>, …::Matrix<…>, 0>>) [879.50ms, 2.79%]
InstantiateFunction(Eigen::internal::call_assignment_no_alias<…::Matrix<…>, …::Product<…>, …::assign_op<…>>) [879.38ms, 2.79%]
InstantiateFunction(Eigen::internal::Assignment<…::Matrix<…>, …::Product<…>, …::assign_op<…>, …::Dense2Dense>::run) [878.87ms, 2.78%]
InstantiateFunction(Eigen::internal::generic_product_impl<…>::evalTo<…::Matrix<…>>) [878.36ms, 2.78%]
InstantiateFunction(Eigen::DenseBase<…::CwiseBinaryOp<…>>::sum) [861.29ms, 2.73%]
InstantiateFunction(Eigen::DenseBase<…::CwiseBinaryOp<…>>::redux<Eigen::internal::scalar_sum_op<double>>) [860.23ms, 2.72%]
InstantiateFunction(Eigen::internal::redux_evaluator<…::CwiseBinaryOp<…>>::redux_evaluator) [843.94ms, 2.67%]
InstantiateFunction(Eigen::internal::evaluator<…::CwiseBinaryOp<…>>::evaluator) [843.87ms, 2.67%]
InstantiateFunction(Eigen::internal::binary_evaluator<…>::binary_evaluator) [843.81ms, 2.67%]
InstantiateFunction(Eigen::internal::binary_evaluator<…>::Data::Data) [843.43ms, 2.67%]
InstantiateFunction(Eigen::internal::evaluator<const Eigen::Transpose<const …::Product<…>>>::evaluator) [843.15ms, 2.67%]
InstantiateFunction(Eigen::internal::evaluator<Eigen::Transpose<const …::Product<…>>>::evaluator) [843.09ms, 2.67%]
InstantiateFunction(Eigen::internal::unary_evaluator<…>::unary_evaluator) [843.04ms, 2.67%]
InstantiateFunction(Eigen::internal::evaluator<const …::Product<…>>::evaluator) [842.89ms, 2.67%]
InstantiateFunction(Eigen::internal::evaluator<…::Product<…>>::evaluator) [842.84ms, 2.67%]
InstantiateFunction(Eigen::internal::product_evaluator<…>::product_evaluator) [842.78ms, 2.67%]
InstantiateFunction(Eigen::internal::generic_product_impl<…>::evalTo<…::Matrix<…>>) [839.68ms, 2.66%]
InstantiateFunction(…::call_assignment_no_alias<…>) [830.19ms, 2.63%]
InstantiateFunction(Eigen::internal::Assignment<…>::run) [829.55ms, 2.63%]
InstantiateFunction(…::call_dense_assignment_loop<…>) [829.46ms, 2.63%]
…4 callee(s) below 2.60%…
```

After Acts#1062 (Delegate @ CKF)

```
├─Frontend [6.56s, 32.18%]
│ ├─PerformPendingInstantiations [3.29s, 16.13%]
│ │ └─…294 callee(s) below 3.70%…
```

- Again, used similar temporal cutoff for fair comparison
  - But to find back CKF::findTrack, we'll need a smaller cutoff

```
├─Frontend [6.56s, 32.18%]
│ ├─PerformPendingInstantiations [3.29s, 16.13%]
│ │ ├─InstantiateFunction(Acts::Test::createMeasurements<…::Propagator<…>, …::SingleCurvilinearTrackParameters<…>>) [600.69ms, 2.95%]
│ │ │ ├─InstantiateFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached>) [564.31ms, 2.77%]
│ │ │ │ └─…15 callee(s) below 2.38%…
│ │ │ └─…7 callee(s) below 2.38%…
│ │ ├─InstantiateFunction(Acts::CombinatorialKalmanFilter<…>::findTracks<…::TestContainerAccessor<…>, …::vector<…>, …::SingleBoundTrackParameters<…>>) [486.94ms, 2.39%]
│ │ │ └─…17 callee(s) below 2.38%…
```

- -87 % decrease → no longer the frontend bottleneck. How ?
  - MeasurementSelector could be moved to a different CU
  - The Eigen pyramid of doom followed it

The Future : Acts#1262

- Acts being integrated in ATLAS → Need to integrate w/ xAOD
  - Need more genericity in trajectory storage → sensitive !

	file	max_rss_main	max_rss_feat	relative
10	CombinatorialKalmanFilterTests.cpp	1087.35	1230.05	113.123
11	GsfTests.cpp	1642.13	1686.95	102.729
12	KalmanFitterTests.cpp	1232.11	1282.43	104.085
13	GainMatrixUpdaterTests.cpp	383.144	437.797	114.264
14	GainMatrixSmootherTests.cpp	329.216	413.655	125.649

- Paul managed to get it down to a ~100MB worst-case cost
  - Where does this come from, and could we do better ?

GainMatrixSmootherTests before

```
Hierarchical profile:
ExecuteCompiler [3.77s, 100.00%]
├─Frontend [2.42s, 64.24%]
│ ├─Source(/opt/spack/opt/spack/…/gcc-12.1.0/boost-1.79.0-pu25xp7wrflbj2xwdb3dcoftqtnde4qg/include/boost/test/unit_test.hpp) [775.90ms, 20.58%]
│ │ ├─Source(/opt/spack/opt/spack/…/gcc-12.1.0/boost-1.79.0-pu25xp7wrflbj2xwdb3dcoftqtnde4qg/include/boost/test/test_tools.hpp) [711.90ms, 18.89%]
│ │ │ └─…9 callee(s) below 15.00%…
│ │ └─…1 callee(s) below 15.00%…
│ ├─PerformPendingInstantiations [750.87ms, 19.92%]
│ │ └─…115 callee(s) below 15.00%…
│ └─…10 callee(s) below 15.00%…
├─Backend [1.11s, 29.57%]
│ ├─Optimizer [789.54ms, 20.95%]
│ │ ├─ModuleInlinerWrapperPass(/mnt/acts/Tests/UnitTests/Core/TrackFitting/GainMatrixSmootherTests.cpp) [636.96ms, 16.90%]
│ │ │ ├─ModuleToPostOrderCGSCCPassAdaptor(/mnt/acts/Tests/UnitTests/Core/TrackFitting/GainMatrixSmootherTests.cpp) [632.76ms, 16.79%]
│ │ │ │ └─…238 callee(s) below 15.00%…
│ │ │ └─…2 callee(s) below 15.00%…
│ │ └─…13 callee(s) below 15.00%…
│ └─…1 callee(s) below 15.00%…
└─…1 callee(s) below 15.00%…
```

GainMatrixSmootherTests after

Hi Eigen !

```
Hierarchical profile:
ExecuteCompiler [5.51s, 100.00%]
├─Frontend [3.15s, 57.07%]
│ ├─PerformPendingInstantiations [1.43s, 25.89%]
│ │ ├─InstantiateFunction(Acts::GainMatrixSmoother::operator()<…::VectorMultiTrajectory>) [708.43ms, 12.86%]
│ │ │ ├─InstantiateFunction(Acts::MultiTrajectory<…>::applyBackwards<(lambda at /mnt/acts/Core/include/Acts/TrackFitting/GainMatrixSmoother.hpp:112:51)>) [688.62ms, 12.50%]
│ │ │ │ ├─InstantiateFunction(Eigen::operator<<<Eigen::Inverse<Eigen::Map<…::Matrix<…>, 0>>>) [677.47ms, 12.29%]
│ │ │ │ │ └─InstantiateFunction(Eigen::DenseBase<Eigen::Inverse<Eigen::Map<…::Matrix<…>, 0>>>::eval) [677.42ms, 12.29%]
│ │ │ │ │   └─InstantiateFunction(Eigen::Matrix<double, 6, 6, 0>::Matrix<Eigen::Inverse<Eigen::Map<…::Matrix<…>, 0>>>) [677.23ms, 12.29%]
│ │ │ │ │     └─InstantiateFunction(Eigen::PlainObjectBase<…::Matrix<…>>::_init1<Eigen::Inverse<…::Map<…>>, Eigen::Inverse<…::Map<…>>>) [676.34ms, 12.27%]
│ │ │ │ │       └─InstantiateFunction(Eigen::PlainObjectBase<Eigen::Matrix<double, 6, 6, 0>>::_set_noalias<Eigen::Inverse<Eigen::Map<…::Matrix<…>, 0>>>) [676.30ms, 12.27%]
│ │ │ │ │         └─InstantiateFunction(Eigen::internal::call_assignment_no_alias<…::Matrix<…>, …::Inverse<…>, …::assign_op<…>>) [676.22ms, 12.27%]
│ │ │ │ │           └─InstantiateFunction(Eigen::internal::Assignment<…::Matrix<…>, …::Inverse<…>, …::assign_op<…>, …::Dense2Dense>::run) [675.96ms, 12.27%]
│ │ │ │ │             └─InstantiateFunction(Eigen::internal::compute_inverse<Eigen::Map<…::Matrix<…>, 0>, Eigen::Matrix<double, 6, 6, 0>, 6>::run) [675.27ms, 12.25%]
│ │ │ │ │               ├─InstantiateFunction(Eigen::MatrixBase<Eigen::Map<…::Matrix<…>, 0>>::partialPivLu) [576.95ms, 10.47%]
│ │ │ │ │               │ └─InstantiateFunction(Eigen::PartialPivLU<…::Matrix<…>>::PartialPivLU<Eigen::Matrix<double, 6, 6, 0>>) [576.87ms, 10.47%]
│ │ │ │ │               │   ├─InstantiateFunction(Eigen::PartialPivLU<…::Matrix<…>>::compute<Eigen::Matrix<double, 6, 6, 0>>) [573.72ms, 10.41%]
│ │ │ │ │               │   │ ├─InstantiateFunction(Eigen::PartialPivLU<…::Matrix<…>>::compute) [567.75ms, 10.30%]
│ │ │ │ │               │   │ │ └─…6 callee(s) below 10.00%…
│ │ │ │ │               │   │ └─…1 callee(s) below 10.00%…
│ │ │ │ │               │   └─…1 callee(s) below 10.00%…
│ │ │ │ │               └─…4 callee(s) below 10.00%…
│ │ │ │ └─…3 callee(s) below 10.00%…
│ │ │ └─…5 callee(s) below 10.00%…
│ │ └─…130 callee(s) below 10.00%…
│ ├─Source(/opt/spack/opt/spack/…/gcc-12.1.0/boost-1.79.0-pu25xp7wrflbj2xwdb3dcoftqtnde4qg/include/boost/test/unit_test.hpp) [749.81ms, 13.61%]
│ │ ├─Source(/opt/spack/opt/spack/…/gcc-12.1.0/boost-1.79.0-pu25xp7wrflbj2xwdb3dcoftqtnde4qg/include/boost/test/test_tools.hpp) [687.67ms, 12.48%]
│ │ │ └─…9 callee(s) below 10.00%…
│ │ └─…1 callee(s) below 10.00%…
│ └─…8 callee(s) below 10.00%…
├─Backend [1.87s, 34.02%]
│ ├─Optimizer [1.27s, 23.09%]
│ │ ├─ModuleInlinerWrapperPass(/mnt/acts/Tests/UnitTests/Core/TrackFitting/GainMatrixSmootherTests.cpp) [975.21ms, 17.70%]
│ │ │ ├─ModuleToPostOrderCGSCCPassAdaptor(/mnt/acts/Tests/UnitTests/Core/TrackFitting/GainMatrixSmootherTests.cpp) [967.32ms, 17.55%]
│ │ │ │ └─…350 callee(s) below 10.00%…
│ │ │ └─…2 callee(s) below 10.00%…
│ │ └─…13 callee(s) below 10.00%…
│ └─CodeGenPasses [599.97ms, 10.89%]
│   └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFitting/GainMatrixSmootherTests.cpp) [598.00ms, 10.85%]
│     └─…65 callee(s) below 10.00%…
└─…1 callee(s) below 10.00%…
```

What does this give us ?

- Now we have a backtrace of template instantiations
  - Walk through it from the bottom
  - Look for dynamic dispatch & outlining opportunities
  - Check runtime impact wrt expected call frequency

Lessons learned

- Compilation can have **bottlenecks** just like execution
  - If you can find those, optimizing is much easier
  - Compile time isn’t a bad proxy for compile RSS

Lessons learned

- Compilation can have **bottlenecks** just like execution
  - If you can find those, optimizing is much easier
  - Compile time isn't a bad proxy for compile RSS
- Mind the run-time vs compile-time **tradeoffs**
  - Outlining or vtable *at the right layer* is often good enough
  - If you can't split code in CUs, you'll die by combinatorics
  - Eigen-style cleverness does not scale → At least isolate it*

* Or, better yet, remove it after checking what you get for the price.

Lessons learned

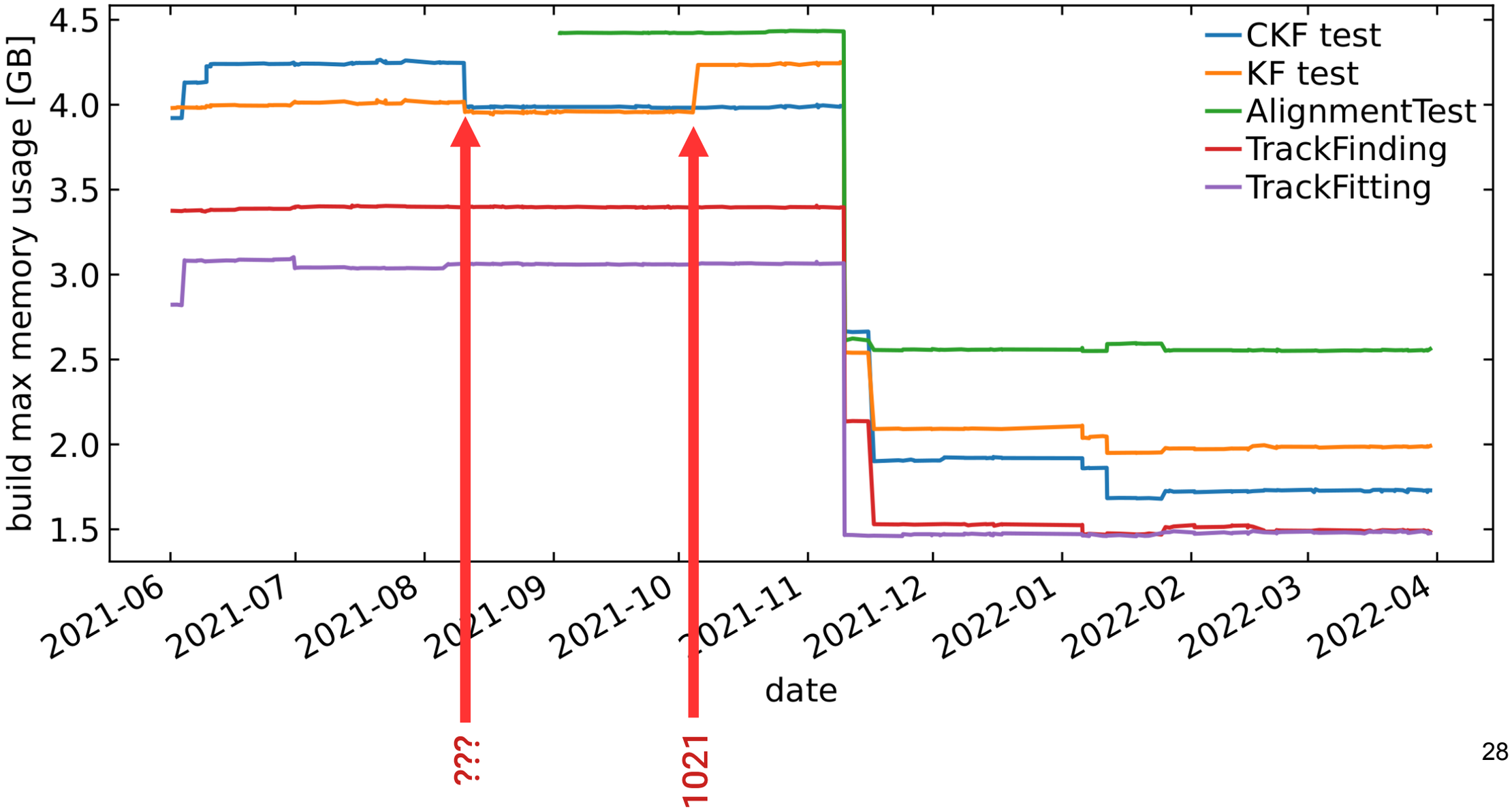
- Compilation can have **bottlenecks** just like execution
  - If you can find those, optimizing is much easier
  - Compile time isn't a bad proxy for compile RSS
- Mind the run-time vs compile-time **tradeoffs**
  - Outlining or vtable *at the right layer* is often good enough
  - If you can't split code in CUs, you'll die by combinatorics
  - Eigen-style cleverness does not scale → At least isolate it*
- Compilation performance can and should be **monitored**

* Or, better yet, remove it after checking what you get for the price.

Paul’s turn !

(Or backup)

Want some more ?



Acts#1021 regression

- This PR adds a reverse filtering option to the KalmanFitter
  - Implemented as yet another template parameter
  - Provided with matching unit tests
- It results in a +200MB build-RSS bump in tests
  - Is this just due to extra testing code, or will it affect users ?

Before Acts#1021

```
Hierarchical profile:
ExecuteCompiler [44.34s, 100.00%]
├─Backend [25.16s, 56.75%]
│ ├─Optimizer [18.05s, 40.71%]
│ │ ├─ModuleInlinerWrapperPass(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [13.49s, 30.42%]
│ │ │ ├─ModuleToPostOrderCGSCCPassAdaptor(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [13.33s, 30.07%]
│ │ │ │ └─…4351 callee(s) below 15.00%…
│ │ │ └─…2 callee(s) below 15.00%…
│ │ └─…18 callee(s) below 15.00%…
│ └─CodeGenPasses [7.10s, 16.00%]
│   └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [7.06s, 15.92%]
│     └─…487 callee(s) below 15.00%…
├─Frontend [12.28s, 27.68%]
│ ├─PerformPendingInstantiations [9.05s, 20.41%]
│ │ └─…254 callee(s) below 15.00%…
│ └─…37 callee(s) below 15.00%…
└─Frontend [6.70s, 15.10%]
  └─…1863 callee(s) below 15.00%…
```

- This time we’re gonna need a higher level view…

After Acts#1021

```
Hierarchical profile:
ExecuteCompiler [54.53s, 100.00%]
├─Backend [32.64s, 59.86%]
  ├─Optimizer [22.98s, 42.14%]
    ├─ModuleInlinerWrapperPass(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [17.56s, 32.20%]
      ├─ModuleToPostOrderCGSCCPassAdaptor(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [17.38s, 31.87%]
        └─…5103 callee(s) below 13.00%…
      └─…2 callee(s) below 13.00%…
    └─…18 callee(s) below 13.00%…
  └─CodeGenPasses [9.64s, 17.67%]
    └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [9.59s, 17.59%]
      └─…515 callee(s) below 13.00%…
├─Frontend [13.91s, 25.51%]
  ├─PerformPendingInstantiations [10.39s, 19.05%]
    └─…278 callee(s) below 13.00%…
  └─…43 callee(s) below 13.00%…
└─Frontend [7.75s, 14.22%]
  └─…2071 callee(s) below 13.00%…
```

+7,5s (30%)

+1,6s (13%)

+1,1s (16%)

- This time we’re gonna need a higher level view…
  - Mostly adds compiler backend work
  - Optimizer hard to analyze (tiny passes) ⟶ check CodeGen

CodeGen before Acts#1021

```
└─CodeGenPasses [7.10s, 16.00%]
  └─OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [7.06s, 15.92%]
    ├─OptFunction(Acts::KalmanFitter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<…>> (…) const) [398.68ms, 0.90%]
    │ ├─RunPass(X86 DAG->DAG Instruction Selection) [96.05ms, 0.22%]
    │ ├─RunPass(Live DEBUG_VALUE analysis) [89.04ms, 0.20%]
    │ └─…56 callee(s) below 0.18%…
    ├─OptFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached> (…) const) [194.46ms, 0.44%]
    │ └─…37 callee(s) below 0.18%…
    ├─OptFunction(Acts::Navigator::target<…::State<…>, …::EigenStepper<…>> (…) const) [180.92ms, 0.41%]
    │ └─…40 callee(s) below 0.18%…
    ├─OptFunction(…::operator() …) [132.25ms, 0.30%]
    │ └─…31 callee(s) below 0.18%…
    ├─OptFunction(Acts::Navigator::status<…::State<…>, …::EigenStepper<…>> (…) const) [127.75ms, 0.29%]
    │ └─…30 callee(s) below 0.18%…
    ├─OptFunction(…::test_method …) [87.25ms, 0.20%]
    │ └─…25 callee(s) below 0.18%…
    ├─OptFunction(Acts::KalmanFitter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<…>> (…) const) [80.46ms, 0.18%]
    │ └─…20 callee(s) below 0.18%…
```

CodeGen after Acts#1021

```
└CodeGenPasses [9.64s, 17.67%]
  └OptModule(/mnt/acts/Tests/UnitTests/Core/TrackFitting/KalmanFitterTests.cpp) [9.59s, 17.59%]
    ├OptFunction(Acts::KalmanFitter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<…>> (…) const) [420.68ms, 0.77%]
    │ ├RunPass(X86 DAG->DAG Instruction Selection) [99.78ms, 0.18%]
    │ ├RunPass(Live DEBUG_VALUE analysis) [92.95ms, 0.17%]
    │ └…66 callee(s) below 0.15%…
    ├OptFunction(Acts::KalmanFitter<…>::Actor<…>::operator()<…::State<…>, …::EigenStepper<…>> (…) const) [419.40ms, 0.77%]
    │ ├RunPass(X86 DAG->DAG Instruction Selection) [104.00ms, 0.19%]
    │ ├RunPass(Live DEBUG_VALUE analysis) [92.75ms, 0.17%]
    │ └…66 callee(s) below 0.15%…
    ├OptFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached> (…) const) [233.58ms, 0.43%]
    │ └…41 callee(s) below 0.15%…
    ├OptFunction(Acts::Propagator<…>::propagate<…::SingleCurvilinearTrackParameters<…>, …::PropagatorOptions<…>, …::PathLimitReached> (…) const) [223.69ms, 0.41%]
    │ └…39 callee(s) below 0.15%…
    ├OptFunction(Acts::Navigator::target<…::State<…>, …::EigenStepper<…>> (…) const) [212.28ms, 0.39%]
    │ └…47 callee(s) below 0.15%…
    ├OptFunction(Acts::Navigator::target<…::State<…>, …::EigenStepper<…>> (…) const) [211.01ms, 0.39%]
    │ └…45 callee(s) below 0.15%…
    ├OptFunction(Acts::Navigator::status<…::State<…>, …::EigenStepper<…>> (…) const) [148.79ms, 0.27%]
    │ └…35 callee(s) below 0.15%…
    ├OptFunction(Acts::Navigator::status<…::State<…>, …::EigenStepper<…>> (…) const) [146.13ms, 0.27%]
    │ └…36 callee(s) below 0.15%…
    ├OptFunction(…::operator() …) [143.24ms, 0.26%]
    │ └…33 callee(s) below 0.15%…
    ├OptFunction(…::test_method …) [100.35ms, 0.18%]
    │ └…25 callee(s) below 0.15%…
    ├OptFunction(Acts::KalmanFitter<…>::Actor<…>::filter<…::State<…>, …::EigenStepper<…>> (…) const) [97.56ms, 0.18%]
    │ └…25 callee(s) below 0.15%…
```

- More complex test → More template instantiations
  - Some of the associated work is duplicated
  - Should not affect production (only 1 KF instantiated there)

The Great August 2021 Mystery

- Monitoring observed a 200-300MB RSS reduction then
- Does not reproduce locally with clang 14 or gcc 12.1
- Timestamp matches PR 905, which seems largely unrelated
- Maybe a change in CI environment ?