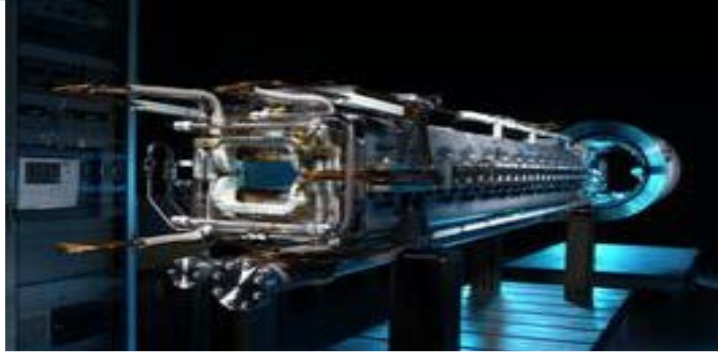
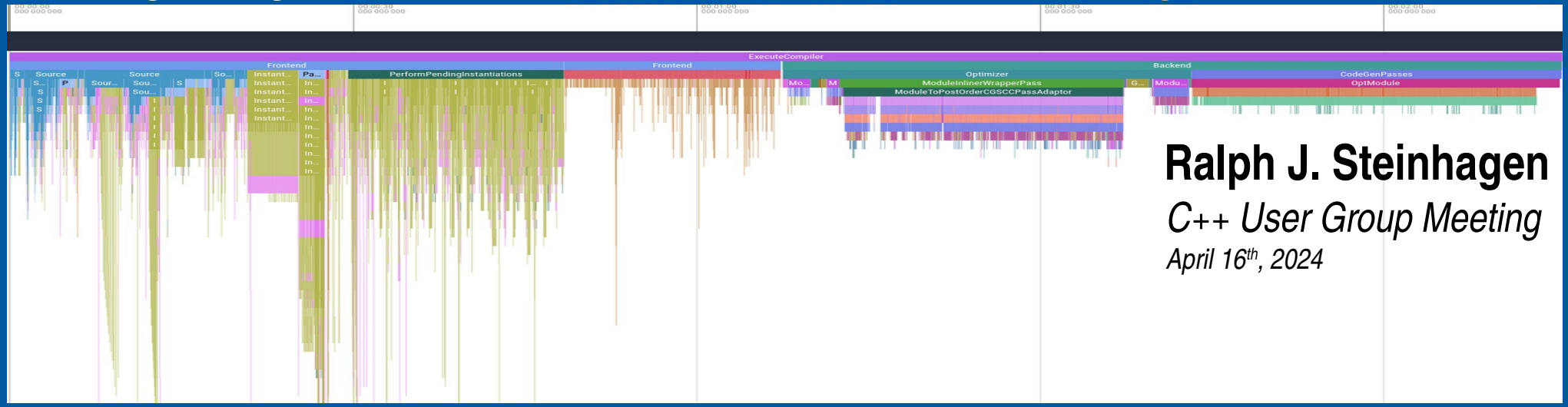




Optimising Build-Times & Meta-Programming …

… using Clang's '`-ftime-trace`' and '`chrome://tracing/`'



Ralph J. Steinhagen

C++ User Group Meeting

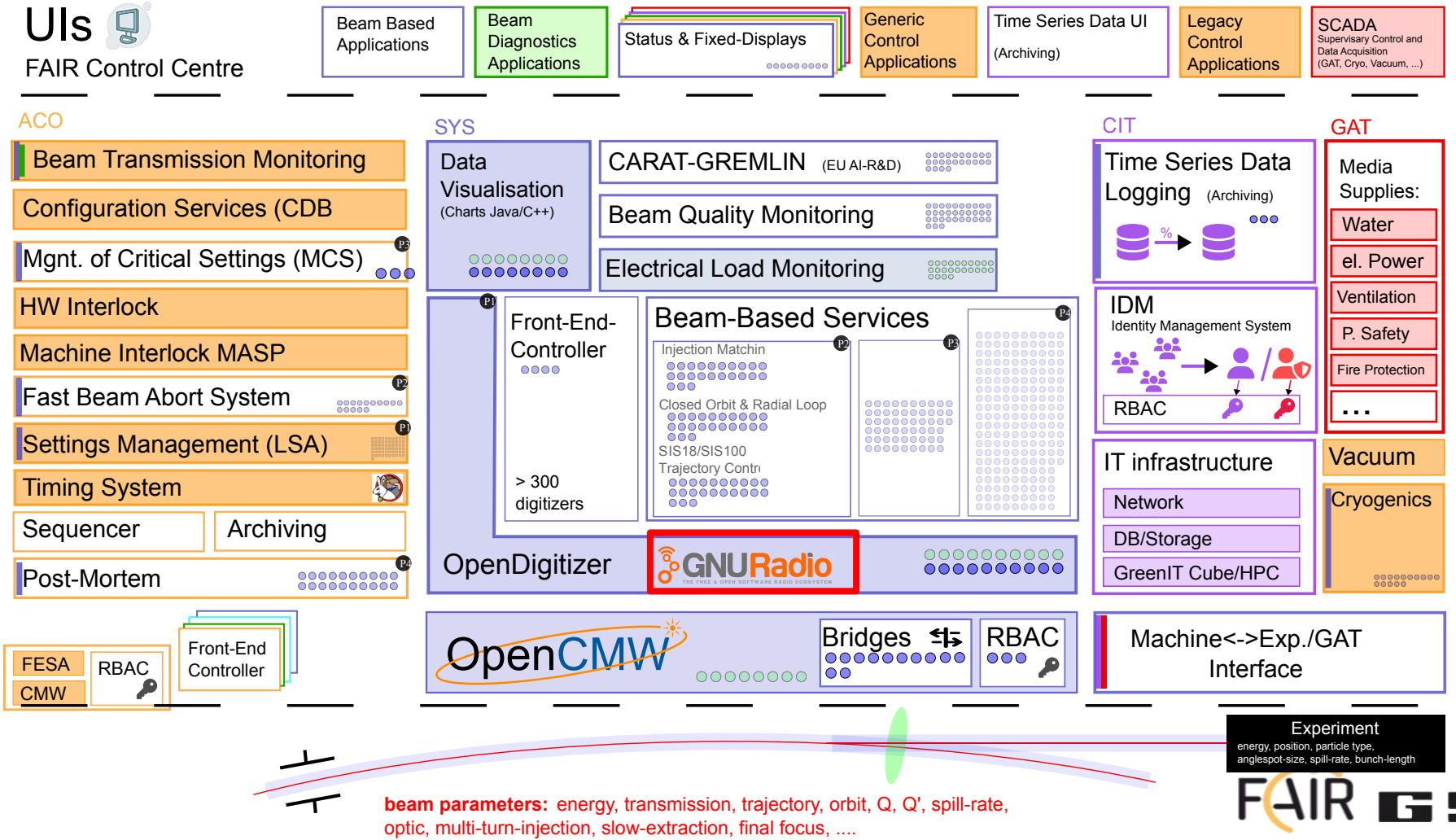
April 16th, 2024

Finland France Germany India Poland Romania Russia Slovenia Sweden UK

GSI HELMHOLTZ ASSOCIATION FAIR

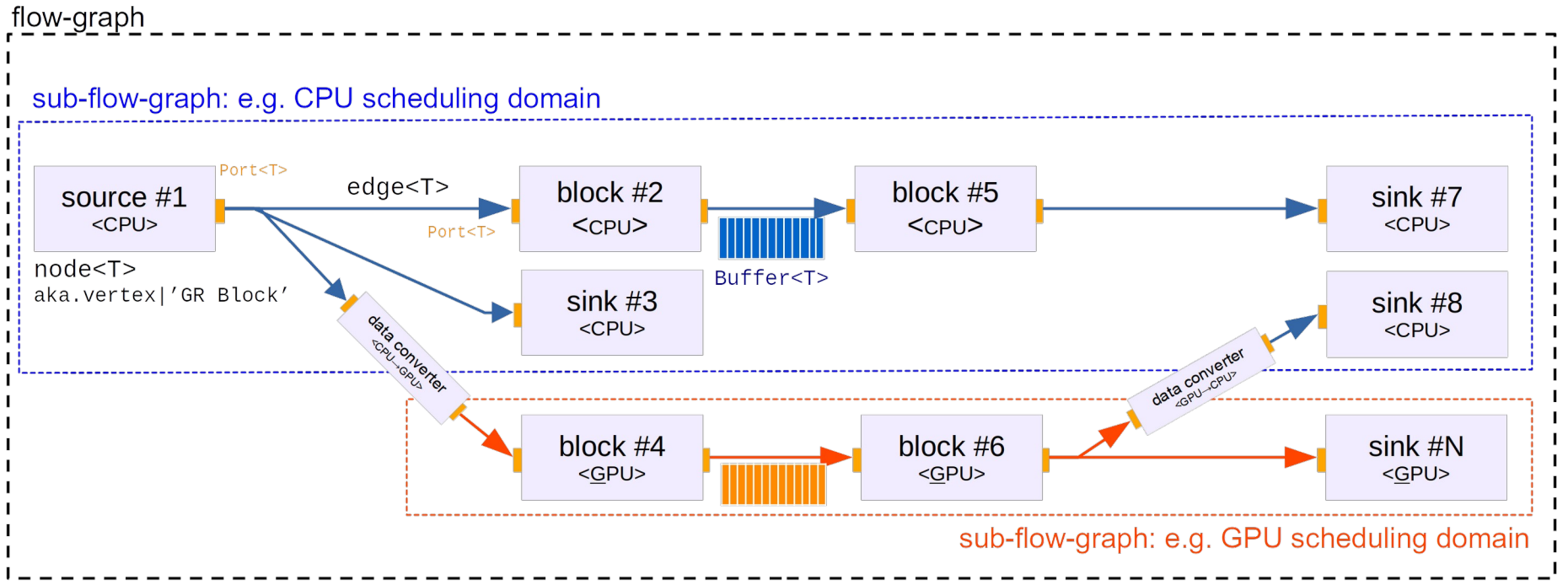
PSP 2.14.17 The 'Big Picture'

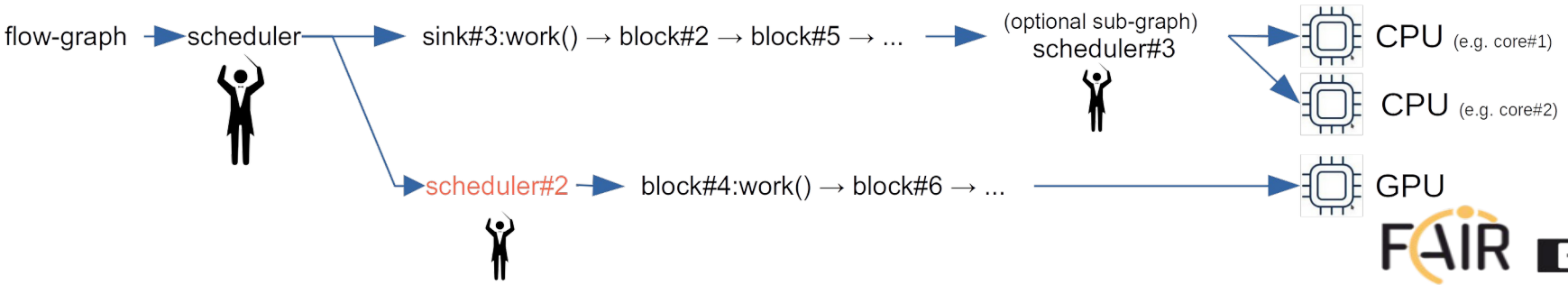
Major Accelerator Control Sub-Systems with SYS Involvement



beam parameters: energy, transmission, trajectory, orbit, Q, Q', spill-rate, optic, multi-turn-injection, slow-extraction, final focus, ...

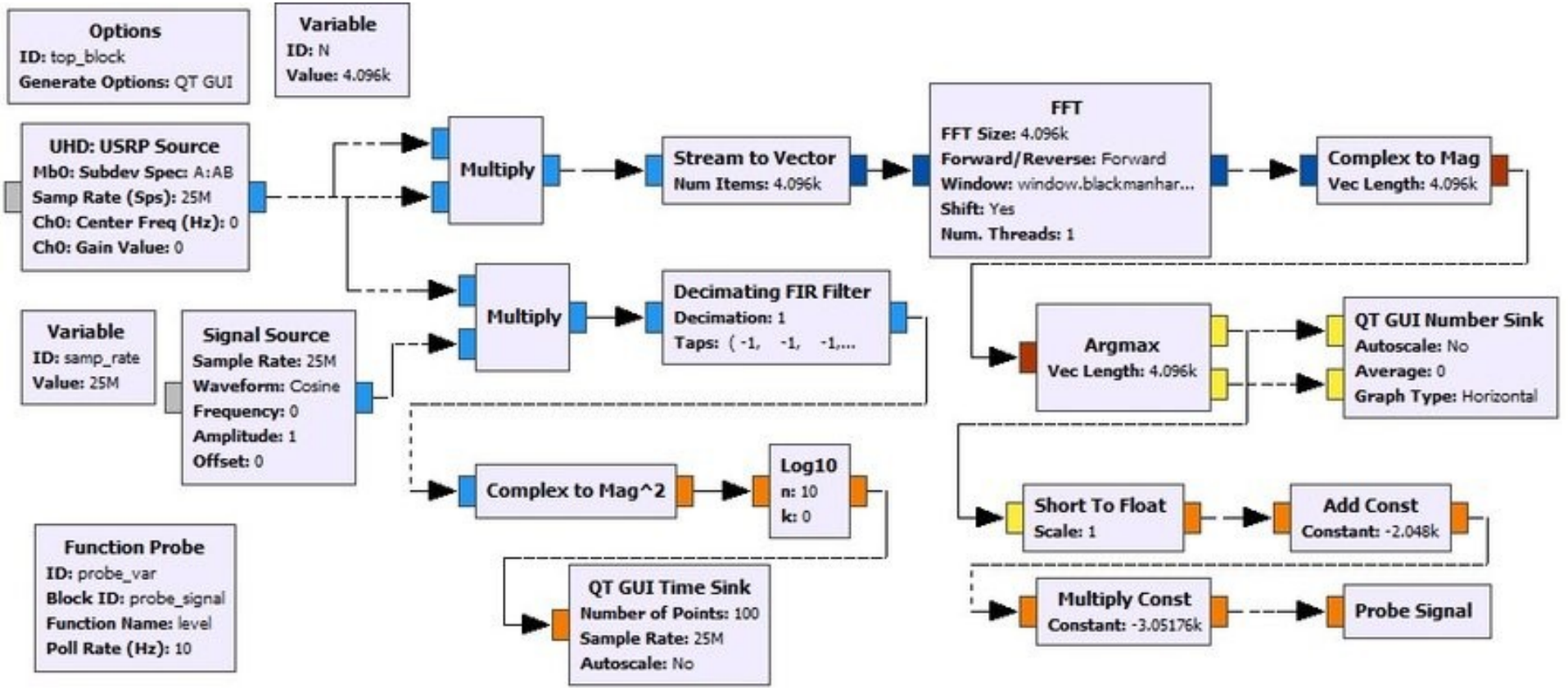
GNU Radio 4.0 – Core Structure





GNU Radio 4.0

Power: Composition with Graph reflecting C++-Signal Processing Flow & Dependencies



Gulzar Khawaja, Wahab Ali & Ozdemir, Ozgur & Guvenc, Ismail. (2017). UAV Air-to-Ground Channel Characterization for mmWave Systems (pdf)

GNU Radio 4.0

Block<T> - Function Signature

```
template<typename T>
struct MyBlock : public gr::Block<MyBlock<T>> {
    MsgPortIn                 msgIn;  /// used for async processing
    MsgPortOut                msgOut;
    std::array<PortIn<T>, 2>  inputs; /// used for sync  processing
    std::array<PortOut<T>, 2> outputs;

    void processMessages(const MsgPortIn &, std::span<const Message> messages) {

        /// asynchronous data processing …

        sendMessage<Command::Notify>(msgOut, /* serviceName */, /* endpoint*/, /* msg*/, /* clientID */);
    }

    template<gr::ConsumableSpan TInSpan, gr::PublishableSpan TOutSpan>
    work::Status processBulk(const std::span<TInSpan> &ins, std::span<TOutSpan> &outs) {

        /// synchronous data processing …

        return gr::work::Status::OK;
    }
};

ENABLE_REFLECTION_FOR_TEMPLATE(MyBlock, msgIn, msgOut, inputs, outputs);
```

GNU Radio 4.0
Block<T> - extensive use of meta-programming/templating

for example: ‘Annotated<>’ Wrapper – containing optional/mandatory meta-info

```
// optional shortening
template<typename U, gr::meta::fixed_string description = "", typename... Arguments>
using A = Annotated<U, description, Arguments...>;

A<float, "sample rate", Visible, Doc<"sampling rate in samples per second (Hz)">>                sample_rate;
A<std::string, "signal name", Doc<"name of the signal">>                                        signal_name;
A<std::string, "signal quantity", Doc<"physical quantity (e.g., 'voltage'). ISO 80000-1:2022.">> signal_quantity;
A<std::string, "signal unit", Doc<"unit of measurement (e.g., '[V]', '[m]').ISO 80000-1:2022">>  signal_unit;
A<float, "signal min,", Doc<"minimum expected signal value">>                                   signal_min;
A<float, "signal max,", Doc<"maximum expected signal value">>                                   signal_max;
```

- improves runtime performance (latency & throughput)
- simplifies user-defined code (N.B. domain-expertise vs. C++/RSE expertise)
- simplifies library-code – DRY principle:
  - compile-time reflection and introspection:
    automatic serialiser, settings management, generate Python-API, in-code documentation …
  - locality: in-code documentation as single source of truth
- Type and Physical-Unit Safety i.e. fail early, ideally during compile-time
  - e.g. quantities: ‘voltage’ vs. ‘energy’ quantity, ‘GeV’ vs. ‘GeV/c’
  - e.g. units: ‘GeV’ vs. ‘MeV’, ‘V` vs. `kV`, ...

FAIR GSI

GNU Radio 4.0 block library of ~600 blocks … from simple to complex

Issue: full project build-times slowly increased to >1h

GNU Radio 4.0

Issue: full project build-times slowly increased to >1h (N.B. 4 min/block → target <20 s/block)

- low-hanging fruits: e.g. Vittorio Romeo, “Improving C++ Compilation Times: Tools & Techniques”, ACCU 2023 (video, slides)

TL;DR:

```
# use ccache if available
find_program(CCACHE_PROGRAM ccache)
if(CCACHE_PROGRAM)
    message(STATUS "Found ccache in ${CCACHE_PROGRAM}")
    set_property(GLOBAL PROPERTY RULE_LAUNCH_COMPILE "${CCACHE_PROGRAM}")
endif()
```

```
cmake -GNinja -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_CXX_FLAGS="-fuse-ld=lld" ..
```

- **gcc13 – default build**
  - real 8m33.364s
  - user 55m57.574s
  - sys 1m51.990s
- **gcc14 – lld + Ninja**
  - real 6m44.549s
  - user 54m41.928s
  - sys 1m21.745s
- **clang18 – default build**
  - real 3m10.854s
  - user 27m43.901s
  - sys 1m3.295s
- **clang18 – lld + Ninja**
  - real 3m11.979s
  - user 27m49.442s
  - sys 1m0.728s

FAIR GSI

Trace-Diagnostic Tooling – Under our very nose ...

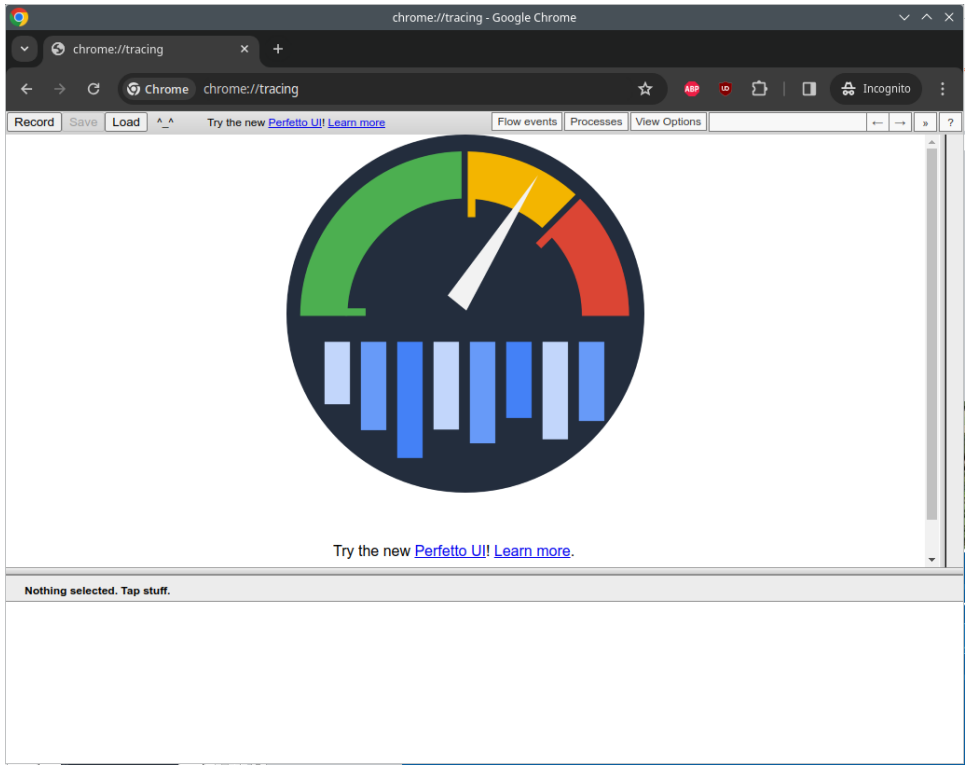
Clang's `-ftime-trace` & `https://github.com/aras-p/ClangBuildAnalyzer/`

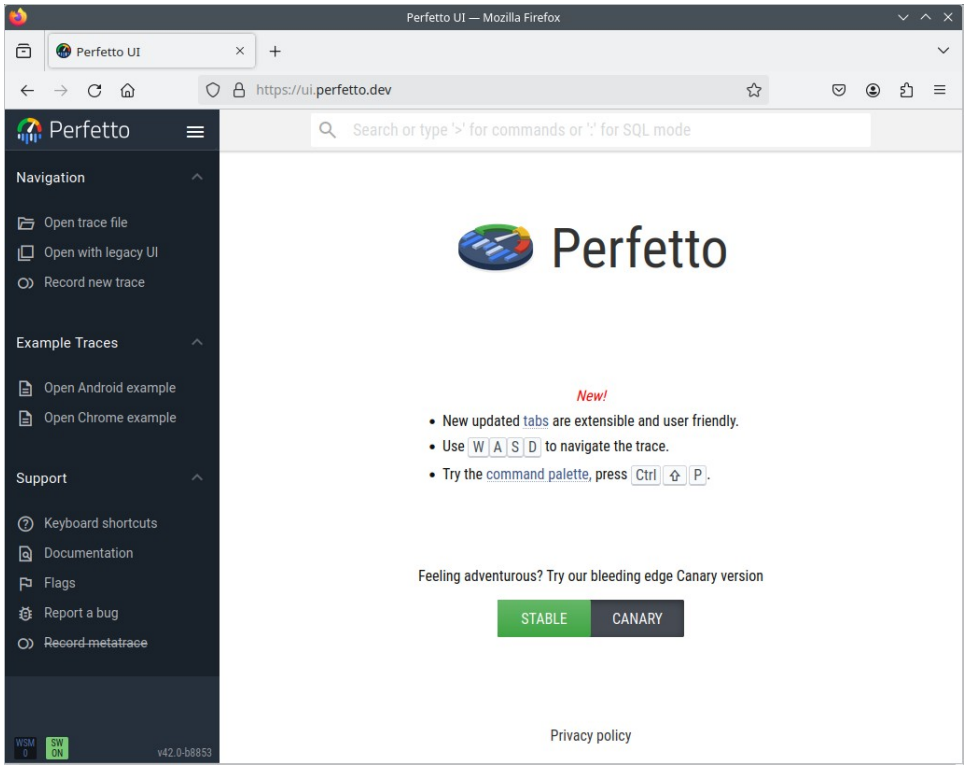
```
cmake -GNinja -DCMAKE_CXX_COMPILER=clang++
      -DCMAKE_CXX_FLAGS="-fuse-ld=lld -ftime-trace"


./ClangBuildAnalyzer --all <artifacts_folder> <capture_file>
   Processing all files and saving to 'analyse.bin'...
   done in 4.2s. Run 'ClangBuildAnalyzer --analyze analyse.bin' to analyze it.
./ClangBuildAnalyzer --analyze <capture_file> > <output.txt>
```

- free & open-source tool developed by Aras Pranckevičius
- **`-ftime-trace`** merged upstream since Clang 9[src]
  - produces Google's Chrome Tracing .json files for each compiled object file
  - no equivalent in GCC or MSVC
- **ClangBuildAnalyser**; parses and aggregates `-ftime-trace` output (~ bin similar results)
  - produces a human-friendly actionable reports

Trace-Diagnostic Tooling – Under our very nose ...

Chrome's chrome://tracing/ (alt: https://ui.perfetto.dev/)

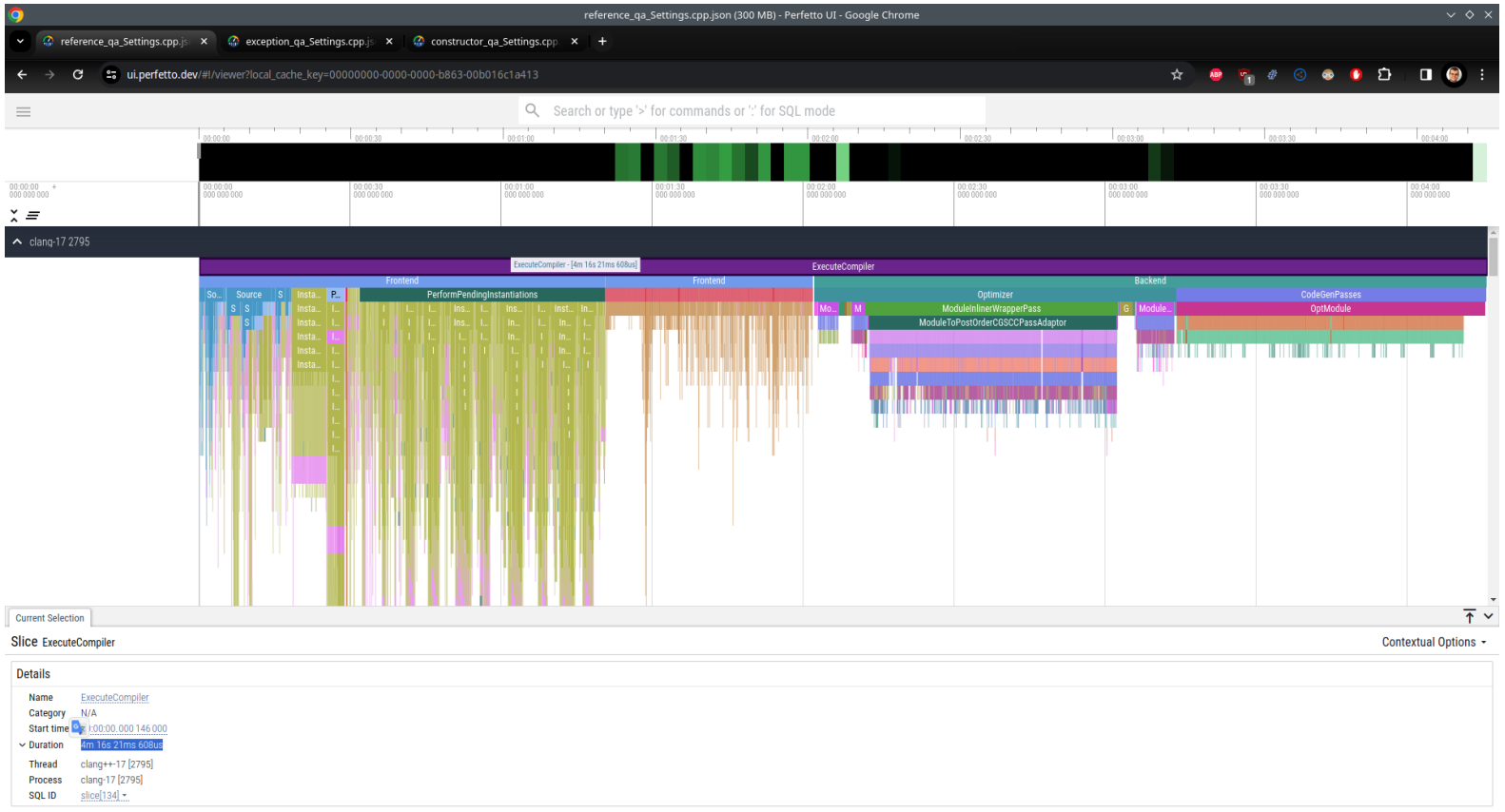




- same format being used by:
  Google, Android, Clang (both: compile- and runtime-analysis), … GR4.0

GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`' – 4min 16s (GR4.0 target: <20s)



GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`'

```
**** Time summary:
Compilation (2 times):
  Parsing (frontend):             67.3 s
  Codegen & opts (backend):       68.1 s
[..]

**** Templates that took longest to instantiate:
  4466 ms: std::visit<(lambda at ../qa_Settings.cpp:63:21), const rva::variant<... (1 times, avg 4466 ms)
  4464 ms: std::__variant_detail::__visitation::__variant::__visit_value<(lambd... (1 times, avg 4464 ms)
  4463 ms: std::__variant_detail::__visitation::__variant::__visit_alt<std::__v... (1 times, avg 4463 ms)
  4462 ms: std::__variant_detail::__visitation::__base::__visit_alt<std::__vari... (1 times, avg 4462 ms)
  4450 ms: std::__variant_detail::__visitation::__base::__make_fmatrix<std::__v... (1 times, avg 4450 ms)
  4448 ms: std::__variant_detail::__visitation::__base::__make_fmatrix_impl<std... (1 times, avg 4448 ms)
  2286 ms: fmt::basic_format_string<char, const std::string &, const rva::varia... (1 times, avg 2286 ms)
  2278 ms: fmt::detail::format_string_checker<char, std::string, rva::variant<s... (1 times, avg 2278 ms)
  2268 ms: fmt::detail::arg_mapper<fmt::basic_format_context<fmt::appender, cha... (1 times, avg 2268 ms)
  2268 ms: fmt::detail::has_const_formatter<rva::variant<std::monostate, bool, ... (1 times, avg 2268 ms)
  2263 ms: fmt::formatter<rva::variant<std::monostate, bool, unsigned char, uns... (1 times, avg 2263 ms)
  2253 ms: std::visit<(lambda at ../cmake-build-d... (1 times, avg 2253 ms)
  2252 ms: std::__variant_detail::__visitation::__variant::__visit_value<(lambd... (1 times, avg 2252 ms)
  2252 ms: std::__variant_detail::__visitation::__variant::__visit_alt<std::__v... (1 times, avg 2252 ms)
  2251 ms: std::__variant_detail::__visitation::__base::__visit_alt<std::__vari... (1 times, avg 2251 ms)
  2250 ms: std::__variant_detail::__visitation::__base::__make_fmatrix<std::__v... (1 times, avg 2250 ms)
  2249 ms: std::__variant_detail::__visitation::__base::__make_fmatrix_impl<std... (1 times, avg 2249 ms)
  2216 ms: gr::meta::tuple_for_each<(lambda at ..... (9 times, avg 246 ms)
  2208 ms: gr::meta::tuple_for_each((lambda at ..... (9 times, avg 245 ms)
[..]
```

GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`'

```
**** Template sets that took longest to instantiate:
  7626 ms: std::__variant_detail::__visitation::__base::__visit_alt<$> (10 times, avg 762 ms)
  7598 ms: std::__variant_detail::__visitation::__base::__make_fmatrix<$> (10 times, avg 759 ms)
  7592 ms: std::__variant_detail::__visitation::__base::__make_fmatrix_impl<$> (10 times, avg 759 ms)
  7565 ms: std::__variant_detail::__visitation::__base::__make_dispatch<$> (1296 times, avg 5 ms)
  7284 ms: std::visit<$> (5 times, avg 1456 ms)
  7280 ms: std::__variant_detail::__visitation::__variant::__visit_value<$> (5 times, avg 1456 ms)
  7277 ms: std::__variant_detail::__visitation::__variant::__visit_alt<$> (5 times, avg 1455 ms)
  6993 ms: std::__variant_detail::__visitation::__base::__dispatcher<$>::__disp... (1296 times, avg 5 ms)
  6593 ms: gr::Block<$>::Block (12 times, avg 549 ms)
  6138 ms: refl::util::for_each<$> (439 times, avg 13 ms)
  5982 ms: refl::util::detail::eval_in_order<$> (439 times, avg 13 ms)
  5518 ms: gr::BlockWrapper<$>::BlockWrapper (8 times, avg 689 ms)
  4523 ms: std::make_unique<$> (27 times, avg 167 ms)
  4404 ms: std::__invokable_r<$> (1365 times, avg 3 ms)
  3825 ms: gr::BasicSettings<$>::BasicSettings (10 times, avg 382 ms)
  3815 ms: std::__variant_detail::__visitation::__variant::__std_visit_exhausti... (1116 times, avg 3 ms)
  3782 ms: std::is_invocable<$> (1198 times, avg 3 ms)
  3734 ms: gr::BlockRegistry::addBlockType<$> (3 times, avg 1244 ms)
  3483 ms: fmt::basic_format_string<$>::basic_format_string<$> (116 times, avg 30 ms)
  2994 ms: gr::meta::tuple_for_each<$> (76 times, avg 39 ms)
  2985 ms: fmt::detail::format_string_checker<$>::format_string_checker (59 times, avg 50 ms)
  2386 ms: fmt::detail::arg_mapper<$>::formattable<$> (15 times, avg 159 ms)
  2382 ms: fmt::detail::has_const_formatter<$> (15 times, avg 158 ms)
  2347 ms: fmt::formatter<$>::format<$> (7 times, avg 335 ms)
  2345 ms: gr::Block<$>::work<$> (8 times, avg 293 ms)
  2236 ms: magic_enum::enum_name<$> (3 times, avg 745 ms)
  2228 ms: std::function<$>::function<$> (21 times, avg 106 ms)
  2219 ms: std::__function::__value_func<$>::__value_func<$> (21 times, avg 105 ms)
  2208 ms: gr::meta::tuple_for_each((lambda at ..... (9 times, avg 245 ms)
  2186 ms: gr::Block<$>::workInternal (8 times, avg 273 ms)
```

FAIR GSI

GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`'

```
**** Functions that took longest to compile:
   275 ms: $_2::operator()() const::'lambda'()::operator()() const (../core/test/qa_Settings.cpp)
   173 ms: $_5::operator()() const::'lambda0'()::operator()() const (../core/test/qa_Settings.cpp)
   145 ms: gr::scheduler::SchedulerBase<gr::scheduler::Simple<(gr::scheduler::E... (../core/test/qa_Settings.cpp)
   139 ms: $_5::__invoke() (../core/test/qa_Settings.cpp)
   113 ms: _ZZN2gr13BasicSettingsINS_12setting_test4SinkIdEEEC1ERS3_ENKUlTyT_E_... (../core/test/qa_Settings.cpp)
   111 ms: gr::Block<gr::setting_test::TestBlock<double>, gr::BlockingIO<true>,... (../core/test/qa_Settings.cpp)
   111 ms: gr::load_grc(gr::PluginLoader&, std::__1::basic_string<char, std::__... (../core/test/qa_Settings.cpp)
    94 ms: gr::Block<gr::setting_test::TestBlock<double>, gr::BlockingIO<true>,... (../core/test/qa_Settings.cpp)
    92 ms: gr::Block<gr::setting_test::Source<double> >::workInternal(unsigned ... (../core/test/qa_Settings.cpp)
    91 ms: _ZZN2gr13BasicSettingsINS_12setting_test9TestBlockIdEEEC1ERS3_ENKUlT... (../core/test/qa_Settings.cpp)
    89 ms: gr::scheduler::Simple<(gr::scheduler::ExecutionPolicy)0, gr::profili... (../core/test/qa_Settings.cpp)
    89 ms: $_2::operator()() const::'lambda5'()::__invoke() (../core/test/qa_Settings.cpp)
    84 ms: gr::Block<gr::setting_test::Sink<double> >::~Block() (../core/test/qa_Settings.cpp)
    83 ms: gr::Block<gr::setting_test::Source<double> >::processMessages(gr::Po... (../core/test/qa_Settings.cpp)
    78 ms: $_2::operator()() const::'lambda3'()::__invoke() (../core/test/qa_Settings.cpp)
    76 ms: $_2::operator()() const::'lambda1'()::__invoke() (../core/test/qa_Settings.cpp)
    76 ms: gr::Block<gr::setting_test::TestBlock<double>, gr::BlockingIO<true>,... (../core/test/qa_Settings.cpp)
    71 ms: _ZZN2gr13BasicSettingsINS_12setting_test9TestBlockIdEEEC1ERS3_ENKUlT... (../core/test/qa_Settings.cpp)
    70 ms: _ZZN2gr13BasicSettingsINS_12setting_test4SinkIdEEEC1ERS3_ENKUlTyT_E_... (../core/test/qa_Settings.cpp)
    70 ms: $_3::operator()() const::'lambda'()::__invoke() (../core/test/qa_Settings.cpp)
    69 ms: $_5::operator()() const::'lambda'()::__invoke() (../core/test/qa_Settings.cpp)
    68 ms: _ZZN2gr13BasicSettingsINS_12setting_test9TestBlockIdEEEC1ERS3_ENKUlT... (../core/test/qa_Settings.cpp)
    67 ms: _ZZN2gr13BasicSettingsINS_12setting_test4SinkIdEEEC1ERS3_ENKUlTyT_E_... (../core/test/qa_Settings.cpp)
[..]
```

FAIR GSI

GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`'

```
**** Function sets that took longest to compile / optimize:
  567 ms: decltype(auto) std::__1::__variant_detail::__visitation::__base::__d... (626 times, avg 0 ms)
  516 ms: gr::Block<$>::workInternal(unsigned long) (8 times, avg 64 ms)
  475 ms: gr::Block<$>::processMessages(gr::Port<$>&, std::__1::span<$>) (8 times, avg 59 ms)
  362 ms: gr::Block<$>::~Block() (10 times, avg 36 ms)
  272 ms: gr::Block<$>::init(std::__1::shared_ptr<$>, std::__1::shared_ptr<$>) (8 times, avg 34 ms)
  253 ms: gr::Block<$>::Block(std::initializer_list<$>) (10 times, avg 25 ms)
  239 ms: std::__1::basic_string<$> gr::meta::type_name<$>() (25 times, avg 9 ms)
  238 ms: gr::lifecycle::StateMachine<$>::changeStateTo(gr::lifecycle::State, ... (11 times, avg 21 ms)
  224 ms: YAML::convert<$>::decode(YAML::Node const&, std::__1::vector<$>&) (11 times, avg 20 ms)
  201 ms: decltype(fp0.out()) fmt::v10::formatter<$>::format<$>(fmt::v10::join... (15 times, avg 13 ms)
  185 ms: char const* refl::descriptor::get_display_name<$>(refl::descriptor::... (66 times, avg 2 ms)
  168 ms: void boost::ext::ut::v2_0_0::reporter<$>::on<$>(boost::ext::ut::v2_0... (19 times, avg 8 ms)
  150 ms: void gr::CircularBuffer<$>::buffer_writer<$>::translate_and_publish<... (10 times, avg 15 ms)
  149 ms: gr::BasicSettings<$>::applyStagedParameters() (10 times, avg 14 ms)
  145 ms: gr::scheduler::SchedulerBase<$>::processScheduledMessages() (1 times, avg 145 ms)
  125 ms: gr::Port<$>::newIoHandler(unsigned long) const (22 times, avg 5 ms)
  119 ms: gr::Port<$>::newTagIoHandler(unsigned long) const (22 times, avg 5 ms)
  117 ms: char const* fmt::v10::formatter<$>::parse<$>(fmt::v10::basic_format_... (8 times, avg 14 ms)
  116 ms: gr::BasicSettings<$>::set(std::__1::map<$> const&, gr::SettingsCtx) (10 times, avg 11 ms)
  113 ms: gr::Block<$>::applyChangedSettings() (8 times, avg 14 ms)
  111 ms: gr::load_grc(gr::PluginLoader&, std::__1::basic_string<$> const&) (1 times, avg 111 ms)
  110 ms: gr::Block<$>::requestStop() (8 times, avg 13 ms)
  102 ms: gr::Port<$>::~Port() (22 times, avg 4 ms)
  101 ms: gr::Block<$>::updatePortsStatus() (8 times, avg 12 ms)
   98 ms: std::__1::pair<$> std::__1::map<$>::insert_or_assign[abi:ue170006]<$... (10 times, avg 9 ms)
   96 ms: decltype(auto) std::__1::__variant_detail::__visitation::__base::__d... (62 times, avg 1 ms)
   92 ms: void fmt::v10::detail::value<$>::format_custom_arg<$>(void*, fmt::v1... (28 times, avg 3 ms)
   89 ms: gr::scheduler::Simple<$>::runSingleThreaded() (1 times, avg 89 ms)
   88 ms: gr::Block<$>::updateInputAndOutputTags(long) (6 times, avg 14 ms)
   88 ms: gr::BasicSettings<$>::get(std::__1::span<$>, gr::SettingsCtx) const (10 times, avg 8 ms)
```

FAIR GSI

GNU Radio 4.0 – Demo Time

Compilation Times – Before using '`-ftime-trace`'

```
**** Expensive headers:
10507 ms: ../core/include/gnuradio-4.0/Block.hpp (included 1 times, avg 10507 ms), included via:
  1x: <direct include>

4703 ms: ../cmake-build-debug-clang17/_deps/ut-src/include/boost/ut.hpp (included 1 times, avg 4703 ms), included via:
  1x: <direct include>

1975 ms: ../core/include/gnuradio-4.0/Graph_yaml_importer.hpp (included 1 times, avg 1975 ms), included via:
  1x: <direct include>

1224 ms: /usr/include/c++/v1/string (included 1 times, avg 1224 ms), included via:
  1x: <direct include>

881 ms: ../core/include/gnuradio-4.0/Scheduler.hpp (included 1 times, avg 881 ms), included via:
  1x: <direct include>

630 ms: ../core/include/gnuradio-4.0/Graph.hpp (included 1 times, avg 630 ms), included via:
  1x: <direct include>

536 ms: ../cmake-build-debug-clang17/_deps/fmt-src/include/fmt/format.h (included 1 times, avg 536 ms), included via:
  1x: <direct include>

215 ms: ../core/include/gnuradio-4.0/Transactions.hpp (included 1 times, avg 215 ms), included via:
  1x: <direct include>

26 ms: ../cmake-build-debug-clang17/_deps/fmt-src/include/fmt/ranges.h (included 1 times, avg 26 ms), included via:
  1x: <direct include>

19 ms: /usr/include/c++/v1/stdexcept (included 1 times, avg 19 ms), included via:
  1x: <direct include>
```

`ClangBuildAnalyzer.ini` Config File (optional)

N.B. should be in invocation root

```
How many of most expensive things are reported?
[counts]
# files that took most time to parse
fileParse = 10
# files that took most time to generate code for
fileCodegen = 10
# functions that took most time to generate code for
function = 30
# header files that were most expensive to include
header = 10
# for each expensive header, this many include paths to it are shown
headerChain = 5
# templates that took longest to instantiate
template = 30

# Minimum times (in ms) for things to be recorded into trace
[minTimes]
# parse/codegen for a file
file = 10

[misc]
# Maximum length of symbol names printed; longer names will get truncated
maxNameLength = 200

# Only print "root" headers in expensive header report,
# i.e. only headers that are directly included by at least one source file
onlyRootHeaders = true
```

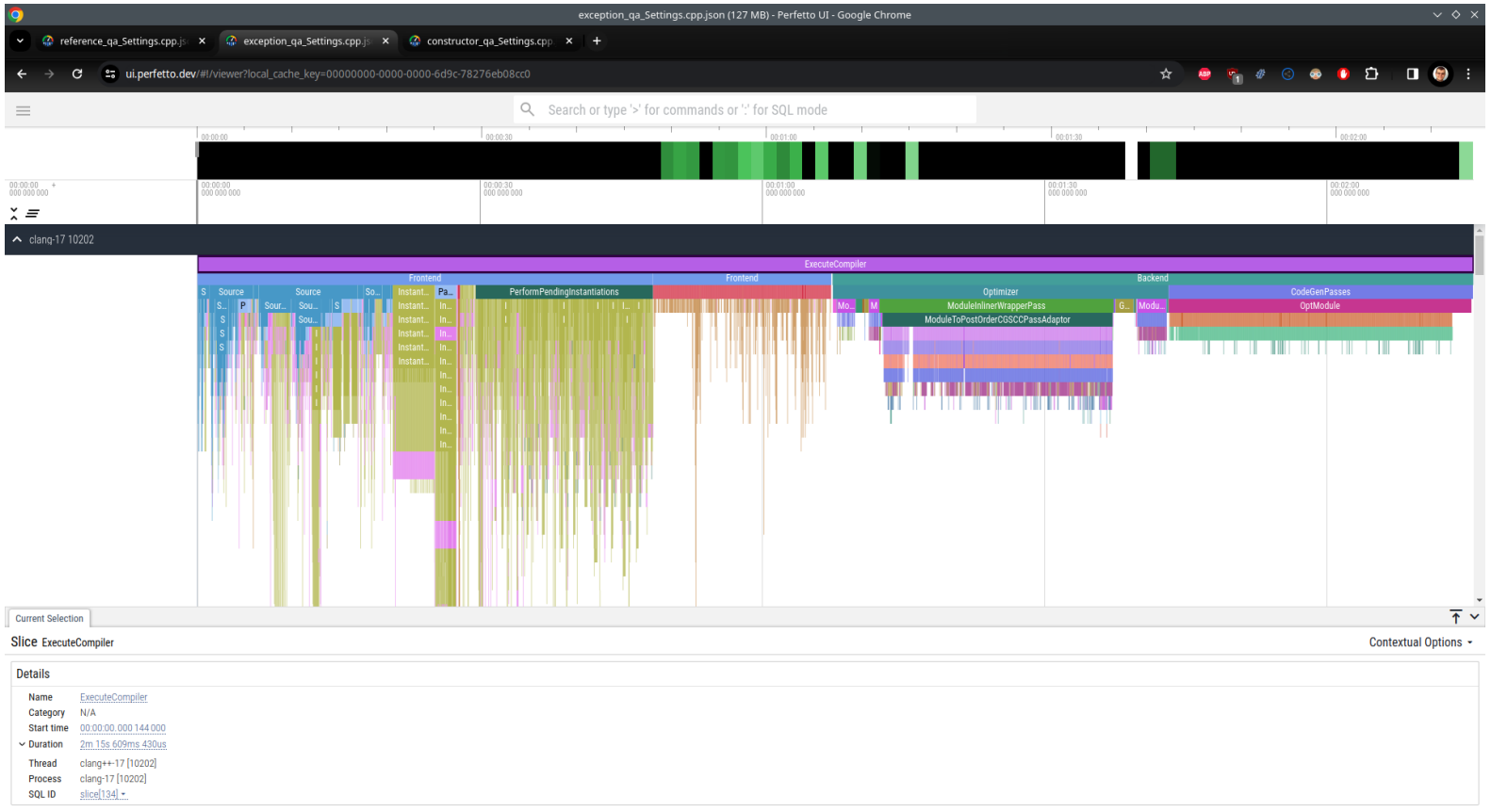
GNU Radio 4.0 – Demo Time
Compilation Times – optimising {fmt}-formatting

```
if (std::string(get_display_name(member)) == key && !std::holds_alternative<Type>(value)) {
    const std::size_t required_index = meta::to_typelist<pmtv::pmt>::index_of<Type>();
    std::visit([&key, &value, &required_index](auto &&arg) {
        using ActualType = std::decay_t<decltype(arg)>;
        throw std::invalid_argument(fmt::format("value for key {} has a wrong type ({}: {}) vs. expected ({}: {})", //
                                                key, value.index(), gr::meta::type_name<ActualType>(),
                                                required_index, gr::meta::type_name<Type>()));
    },value);
}
```

```
if (fieldName == key && !std::holds_alternative<Type>(value)) {
    throw std::invalid_argument([&key, &value] { // lazy evaluation
        const std::size_t actual_index   = value.index();
        const std::size_t required_index = meta::to_typelist<pmtv::pmt>::index_of<Type>(); //
        return fmt::format("value for key '{}' has a wrong type. Index of actual type: {} ({}), Index of expected type: {} ({})",
                           key, actual_index, "<missing pmt type>",
                           required_index, gr::meta::type_name<Type>());
        }());
}
```

GNU Radio 4.0 – Demo Time

Compilation Times – optimising {fmt}-formatting – 2min 15s (GR4.0 target: <20s)



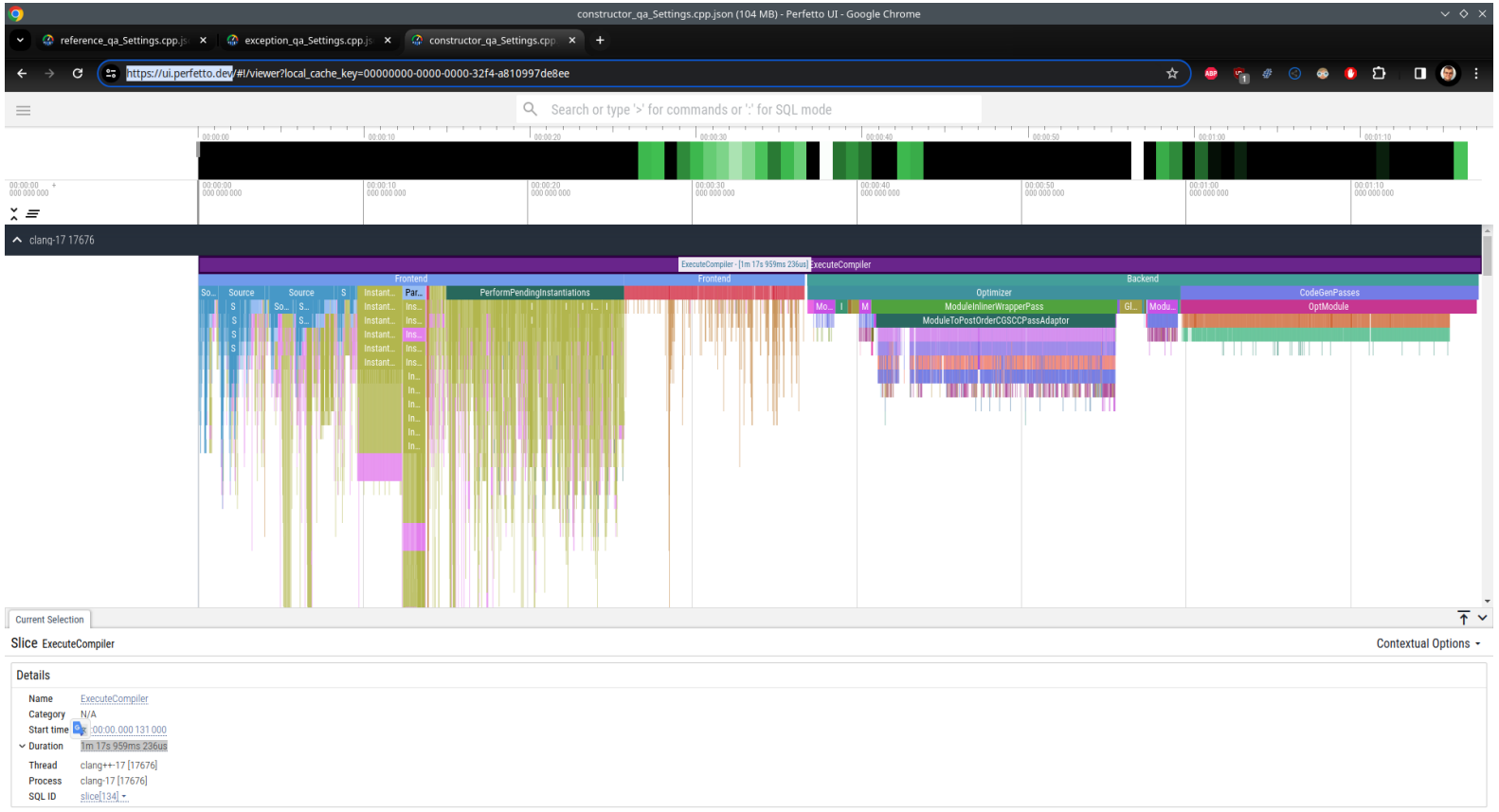
GNU Radio 4.0 – Demo Time
Compilation Times – optimising {fmt} & Settings Handling

Commit: https://github.com/fair-acc/graph-prototype/pull/299/files

- **Shortened `Template<typename …, NTTP...>` parameter names**
  - N.B. compiler seems to work with unmangled names)
- **Flattened Tuple Iteration:**
  - Removed nested `meta::tuple_for_each` loops.
  - Simplified handling with direct metadata checks and additions.
- **Meta-Information Enhancement**:
  - Introduced `hasMetaInfo` concept for block-level meta presence
  - Direct addition of descriptions, documentation, unit, and visibility to metadata.
- **Simplified Error Handling**
  - Refactored error logic using lambda for lazy evaluation.
  - Improved readability and efficiency in type mismatch handling.
- **Clearer Member Handling:**
  - Metadata and settings update logic separated from type conditions.
  - More declarative approach with explicit type safety checks (`constexpr` and static assertions).
- **Performance Gains:**
  - Reduced compile times by optimizing metadata processing and reducing template complexity.

GNU Radio 4.0 – Demo Time

Compilation Times – optimising {fmt} & Settings Handling – 1min 18s (GR4.0 target: <20s)



Live Demo – if time permits I/III

```cpp
// small example to illustrate compile-time analysis
// compile with:
//
// clang++ -std=c++23 -O3 -ftime-trace -o example0 example0.cpp
//

template <unsigned int n>
struct Factorial {
    enum { value = n * Factorial<n - 1>::value };
};

template <>
struct Factorial<0> {
    enum { value = 1 };
};

int main() {
    return Factorial<1000>::value;
}
```

Live Demo – if time permits II/III – adding some includes

```cpp
// small example to illustrate compile-time analysis
// compile with:
//
// clang++ -std=c++23 -O3 -ftime-trace -o example1 example1.cpp
//
#include <cmath>
#include <cstdint>
#include <print>
#include <tuple>
#include <variant>
#include <vector>

template <unsigned int n>
struct Factorial {
    enum { value = n * Factorial<n - 1>::value };
};

template <>
struct Factorial<0> {
    enum { value = 1 };
};

int main() {
    return Factorial<1000>::value;
}
```

Live Demo – if time permits III/III – evil example

```cpp
// small example to illustrate compile-time analysis
// compile with:
//
// clang++ -std=c++23 -O3 -ftime-trace -o example example.cpp
//
#include <bitset>
// and many more include

using Pmt =
        std::variant<std::monostate, bool, std::uint8_t, std::uint16_t,
                     std::uint32_t, std::uint64_t, std::int8_t, std::int16_t,
                     std::int32_t, std::int64_t, float, double>;

// visitor that applies an operation to each variant type
struct OperationVisitor {
    void operator()(const std::monostate &i) const { /* do nothing */ }
    void operator()(bool &i) const { i = !i; }
    void operator()(std::uint8_t &i) const { i = std::pow(i, 2); }
    void operator()(std::uint16_t &i) const { i = std::pow(i, 2); }
    void operator()(std::uint32_t &i) const { i = std::pow(i, 2); }
    void operator()(std::uint64_t &i) const { i = std::pow(i, 2); }
    void operator()(std::int8_t &i) const { i = std::pow(i, 2); }
    void operator()(std::int16_t &i) const { i = std::pow(i, 2); }
    void operator()(std::int32_t &i) const { i = std::pow(i, 2); }
    void operator()(std::int64_t &i) const { i = std::pow(i, 2); }
    void operator()(float &f) const { f = std::sin(f); }
    void operator()(double &d) const { d = std::cos(d); }
};

template <typename T>
void applyOperations(std::vector<T> &variants) {
    for (auto &var : variants) {
        std::visit(OperationVisitor{}, var);
    }
}

template <typename T>
void printVariantVector(std::vector<T> &variants) {
    for (const auto &num : variants) {
        std::visit([](const auto &value) { std::print("value: {}\n", value); },
                   num);
    }
    std::println("- done");
}

template <unsigned int n>
struct Factorial {
    enum { value = n * Factorial<n - 1>::value };
};

template <>
struct Factorial<0> {
    enum { value = 1 };
};

int main() {
    // Example usage
    std::vector<Pmt> vars{{}, true, static_cast<std::uint8_t>(1), static_cast<std::uint16_t>(2),
                          static_cast<std::uint32_t>(3), 4.0f, 5.0, static_cast<std::uint64_t>(6)};

    applyOperations(vars);

    printVariantVector(vars);

    return Factorial<1000>::value;
}
```

Bonus - Trace-Diagnostic Tooling – Under our very nose ...
Clang's XRay Runtime Performance Profiler

TL;DR: using Clang and Debugging with XRay ...

```
cmake -GNinja -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_CXX_FLAGS="-fxray-instrument" ..
```

or:

```
clang++ -o sample -O3 <source> <other flags> \
        -fxray-instrument -fxray-instruction-threshold=1
XRAY_OPTIONS="patch_premain=true xray_mode=xray-basic" ./sample
```

converting this to Google's Trace format:

```
llvm-xray convert --output-format=trace_event --sort --symbolize
                  --instr_map=<binary file name> <xray-log.*> -o <JSON file name>
```

creating a visual call graph:

```
llvm-xray graph xray-log.sample.* -m sample --color-edges=sum --edge-label=sum \
    | unflatten -f -l10 | dot -Tsvg -o sample.svg
```

Bonus - Trace-Diagnostic Tooling – Under our very nose ...
Clang's XRay Runtime Performance Profiler

```cpp
// sample.cpp
// clang++ -o sample -O3 sample.cc -std=c++23 -fxray-instrument -fxray-instruction-threshold=1
// XRAY_OPTIONS="patch_premain=true xray_mode=xray-basic" ./sample
#include <iostream>
#include <thread>

[[clang::xray_always_instrument]] void f() {
    std::cerr << '.';
}

[[clang::xray_always_instrument]] void g() {
    for (int i = 0; i < 1 << 10; ++i) {
        std::cerr << '-';
    }
}

int main(int argc, char* argv[]) {
    std::thread t1([] {
        for (int i = 0; i < 1 << 10; ++i)
            f();
    });
    std::thread t2([] {
        g();
    });
    t1.join();
    t2.join();
    std::cerr << '\n';
}
```