C++23

Modules, std::stacktrace, std::expected

Andi Skrgat

Content of this deck

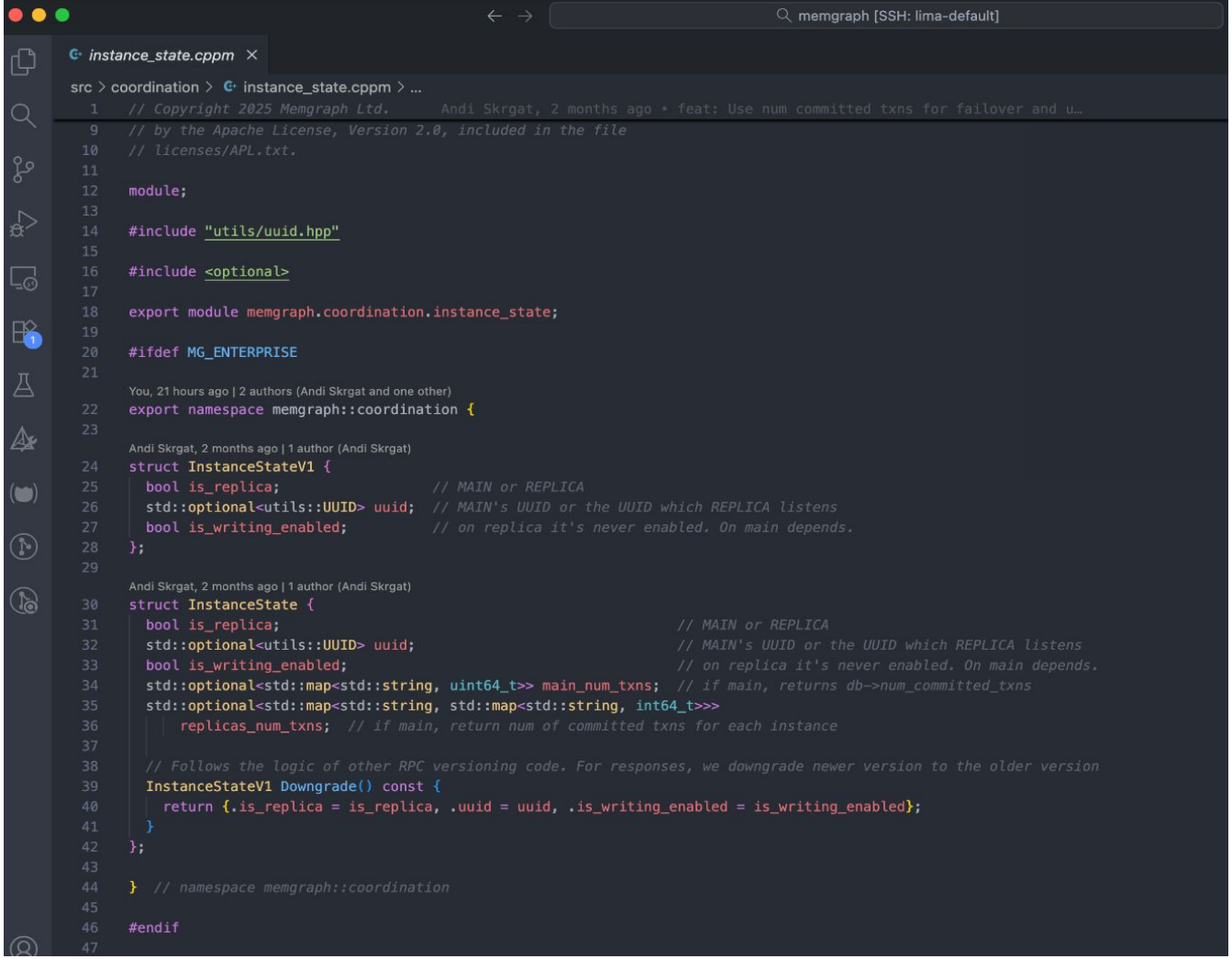
1. Modules
2. std::stacktrace
3. std::expected
4. Practical

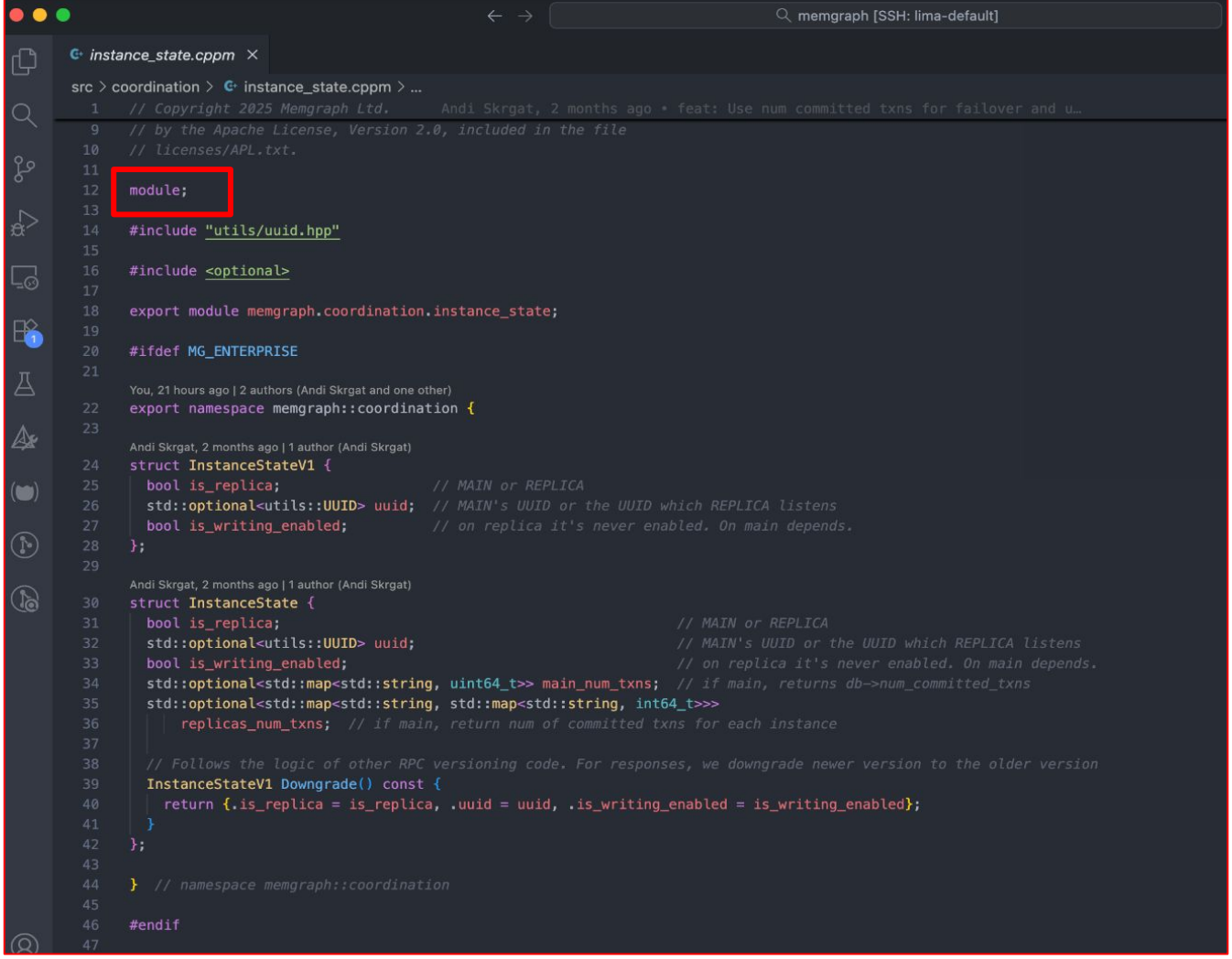
01

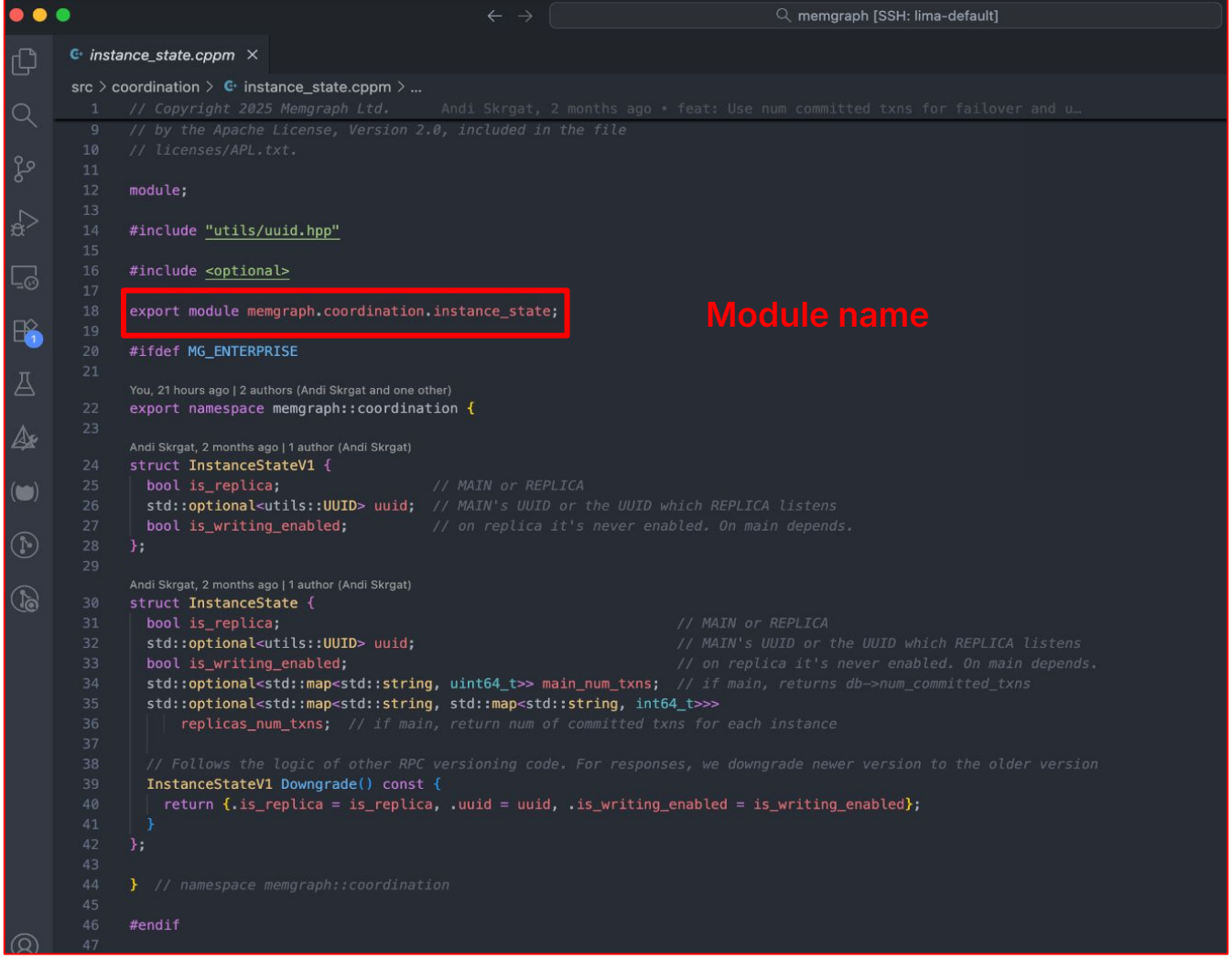
Modules

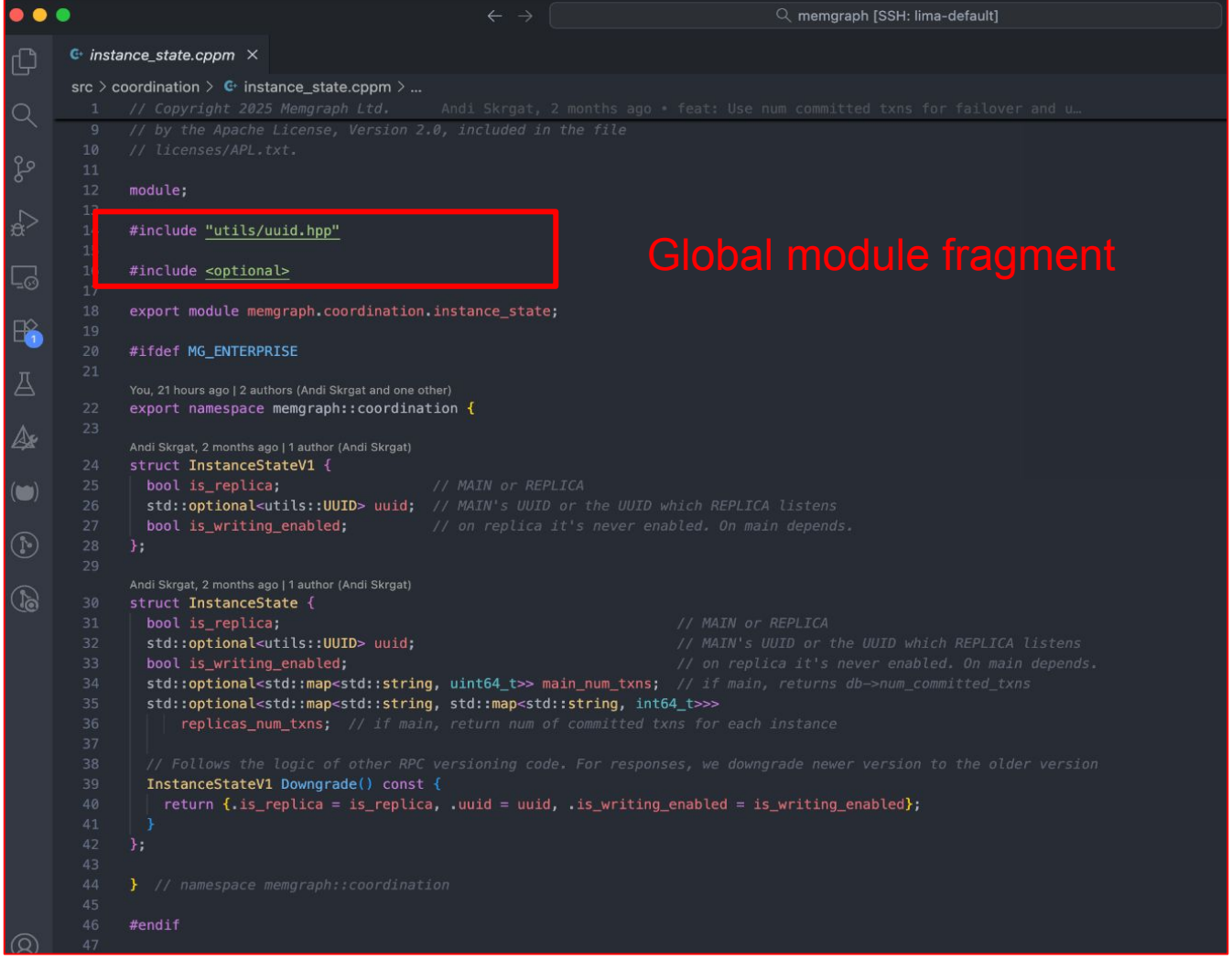
Intro to modules

- A language feature to share declarations and definitions across translation unit
- Since C++20 but good support is fairly recent (CMake removed experimental support this year)
- Completely orthogonal to namespaces


memgraph [SSH: lima-default]
instance_state.cppm
src > coordination > instance_state.cppm > ...
// Copyright 2025 Memgraph Ltd.
Andi Skrgat, 2 months ago • feat: Use num committed txns for failover and u…
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include "utils/uuid.hpp"
#include <optional>
export module memgraph.coordination.instance_state;
#ifdef MG_ENTERPRISE
You, 21 hours ago | 2 authors (Andi Skrgat and one other)
export namespace memgraph::coordination {
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
} // namespace memgraph::coordination
#endif

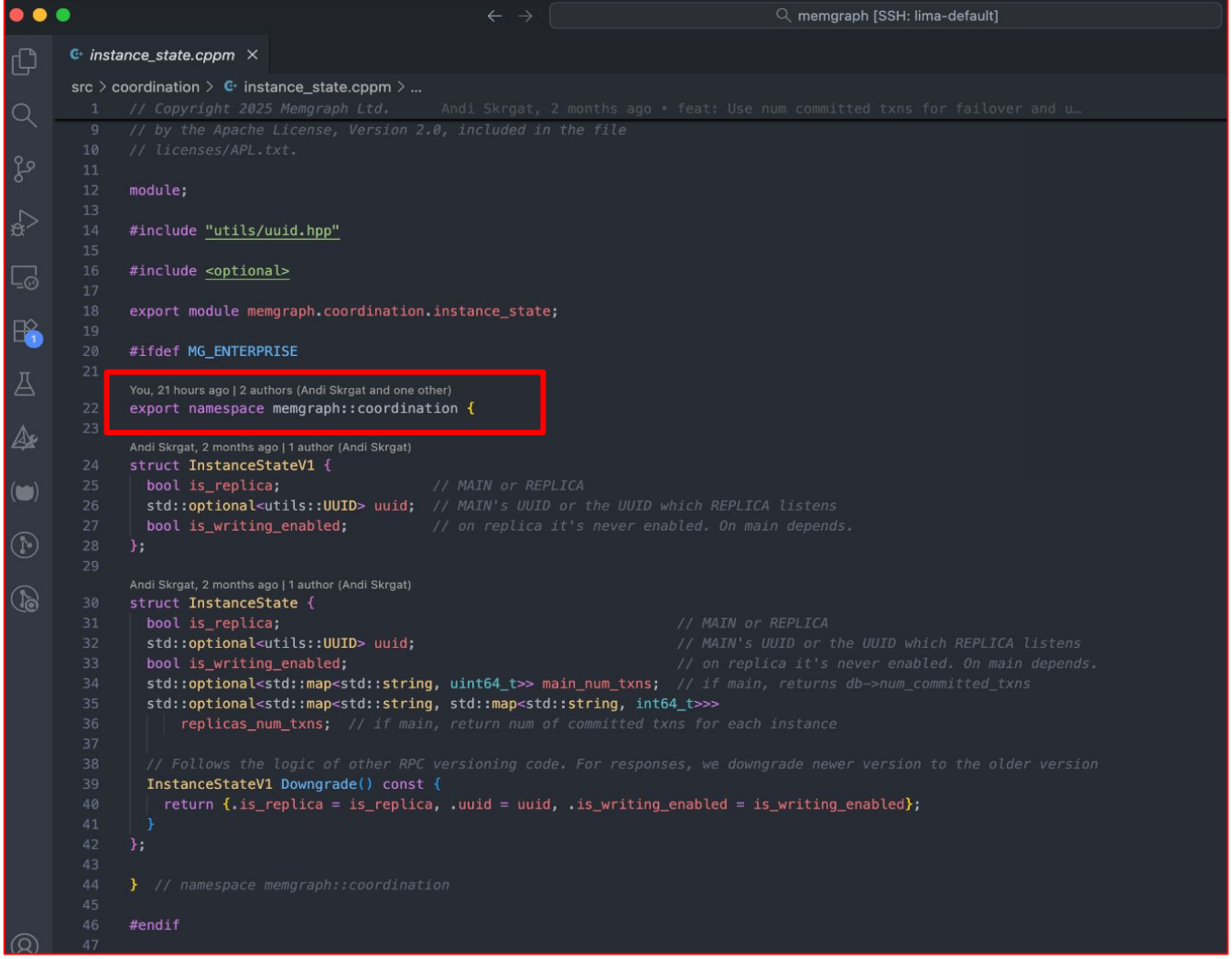

memgraph [SSH: lima-default]
instance_state.cppm
src > coordination > instance_state.cppm > ...
// Copyright 2025 Memgraph Ltd.
Andi Skrgat, 2 months ago • feat: Use num committed txns for failover and u…
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include "utils/uuid.hpp"
#include <optional>
export module memgraph.coordination.instance_state;
#ifdef MG_ENTERPRISE
You, 21 hours ago | 2 authors (Andi Skrgat and one other)
export namespace memgraph::coordination {
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
} // namespace memgraph::coordination
#endif

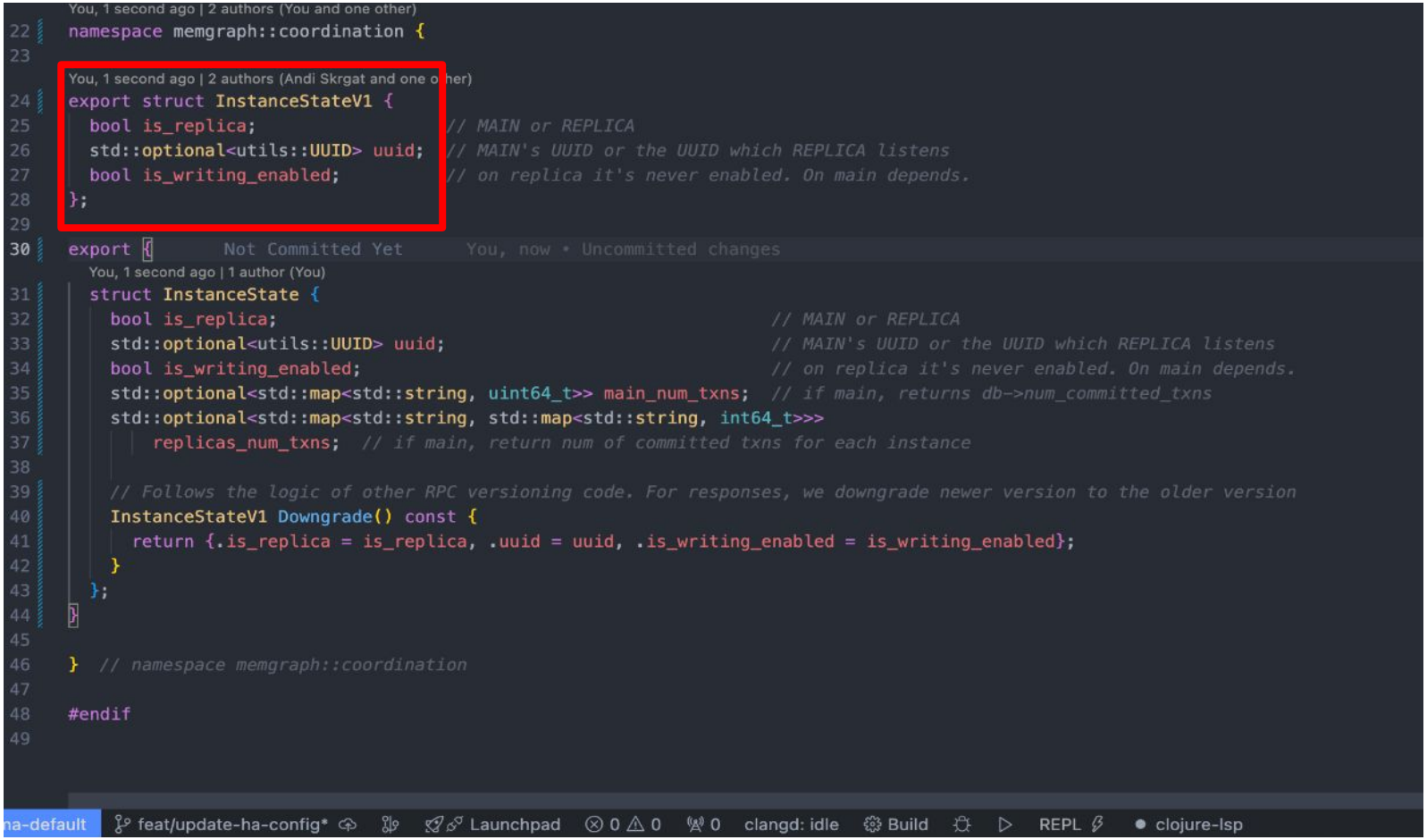

memgraph [SSH: lima-default]
instance_state.cppm
src > coordination > instance_state.cppm > ...
// Copyright 2025 Memgraph Ltd.
Andi Skrgat, 2 months ago • feat: Use num committed txns for failover and u…
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include "utils/uuid.hpp"
#include <optional>
export module memgraph.coordination.instance_state;
Module name
#ifdef MG_ENTERPRISE
You, 21 hours ago | 2 authors (Andi Skrgat and one other)
export namespace memgraph::coordination {
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
} // namespace memgraph::coordination
#endif

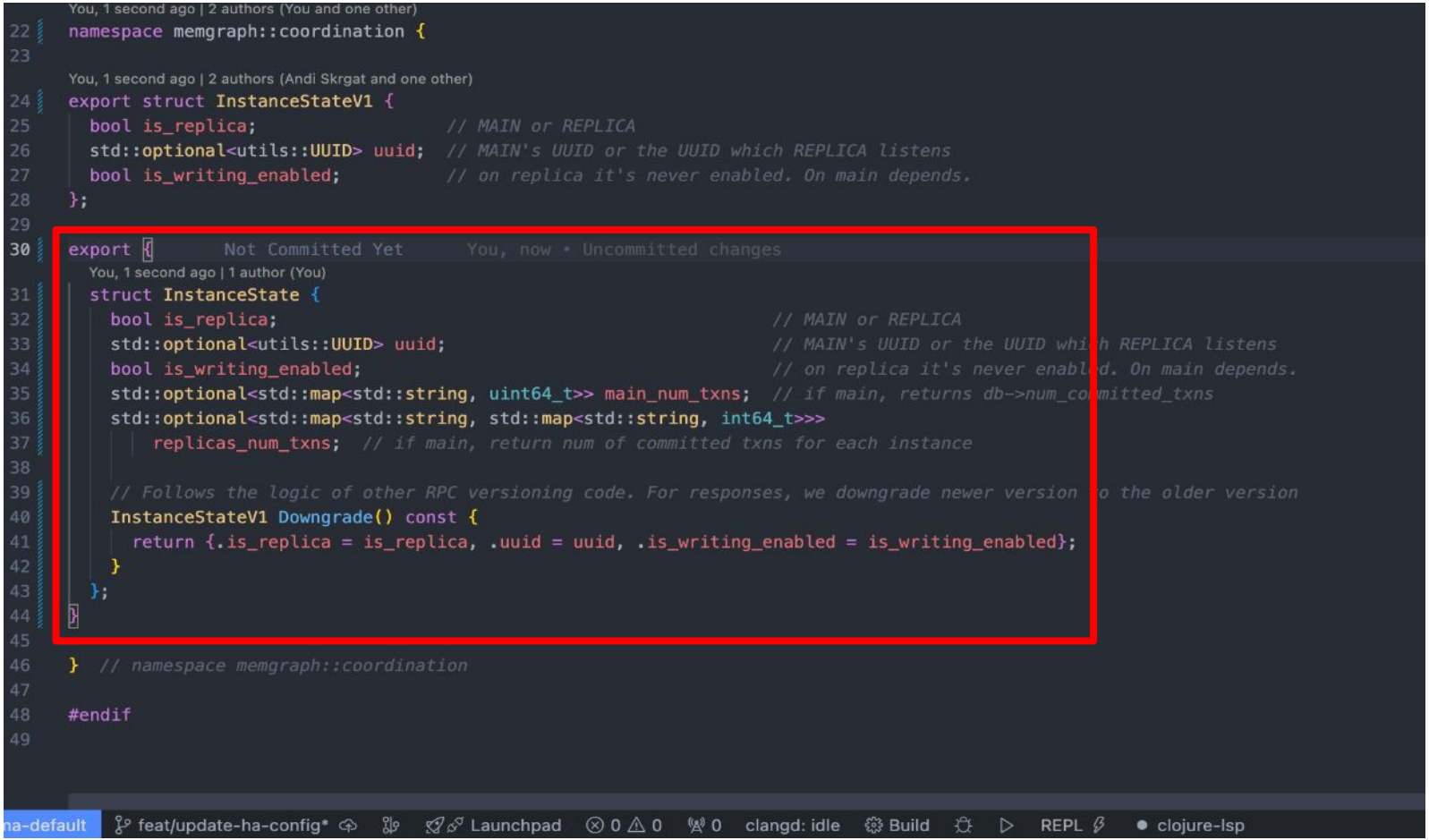

memgraph [SSH: lima-default]
instance_state.cppm
src > coordination > instance_state.cppm > ...
// Copyright 2025 Memgraph Ltd.
Andi Skrgat, 2 months ago • feat: Use num committed txns for failover and u…
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include "utils/uuid.hpp"
#include <optional>
export module memgraph.coordination.instance_state;
#ifdef MG_ENTERPRISE
You, 21 hours ago | 2 authors (Andi Skrgat and one other)
export namespace memgraph::coordination {
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
} // namespace memgraph::coordination
#endif
Global module fragment

Global module fragment

- Used to include headers when importing modules isn’t supported
- It must come at the beginning of the file


memgraph [SSH: lima-default]
instance_state.cppm
src > coordination > instance_state.cppm > ...
// Copyright 2025 Memgraph Ltd.
Andi Skrgat, 2 months ago • feat: Use num committed txns for failover and u…
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include "utils/uuid.hpp"
#include <optional>
export module memgraph.coordination.instance_state;
#ifdef MG_ENTERPRISE
You, 21 hours ago | 2 authors (Andi Skrgat and one other)
export namespace memgraph::coordination {
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
} // namespace memgraph::coordination
#endif


You, 1 second ago | 2 authors (You and one other)
namespace memgraph::coordination {
You, 1 second ago | 2 authors (Andi Skrgat and one other)
export struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
export {
Not Committed Yet
You, now • Uncommitted changes
You, 1 second ago | 1 author (You)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
}
} // namespace memgraph::coordination
#endif
ha-default
feat/update-ha-config*
Launchpad
0 0
0
clangd: idle
Build
REPL
clojure-lsp

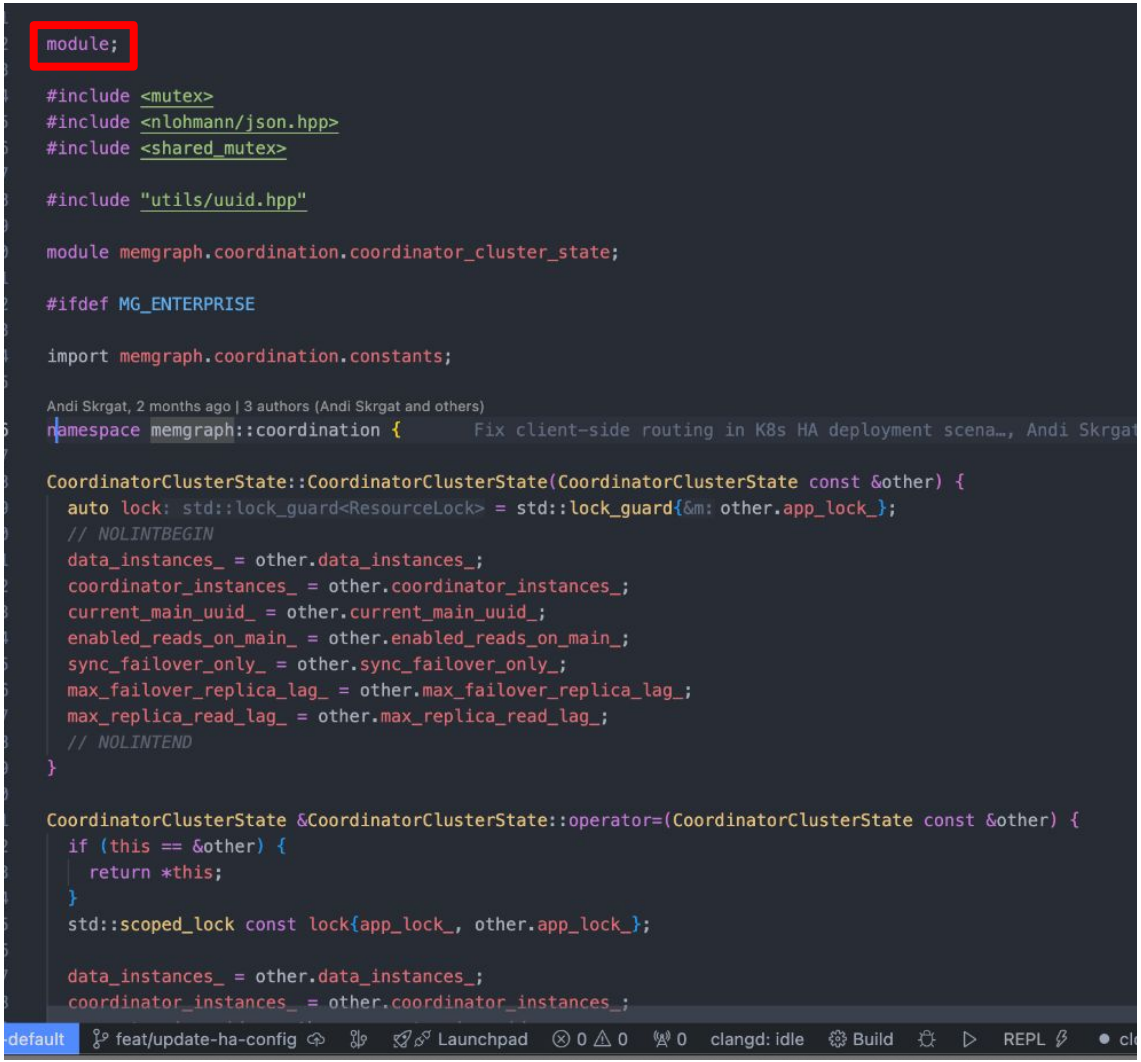

You, 1 second ago | 2 authors (You and one other)
namespace memgraph::coordination {
You, 1 second ago | 2 authors (Andi Skrgat and one other)
export struct InstanceStateV1 {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
};
export {
Not Committed Yet
You, now • Uncommitted changes
You, 1 second ago | 1 author (You)
struct InstanceState {
bool is_replica; // MAIN or REPLICA
std::optional<utils::UUID> uuid; // MAIN's UUID or the UUID which REPLICA listens
bool is_writing_enabled; // on replica it's never enabled. On main depends.
std::optional<std::map<std::string, uint64_t>> main_num_txns; // if main, returns db->num_committed_txns
std::optional<std::map<std::string, std::map<std::string, int64_t>>>
replicas_num_txns; // if main, return num of committed txns for each instance
// Follows the logic of other RPC versioning code. For responses, we downgrade newer version to the older version
InstanceStateV1 Downgrade() const {
return {.is_replica = is_replica, .uuid = uuid, .is_writing_enabled = is_writing_enabled};
}
};
}
} // namespace memgraph::coordination
#endif
ha-default
feat/update-ha-config*
Launchpad
0 0
0
clangd: idle
Build
REPL
clojure-lsp

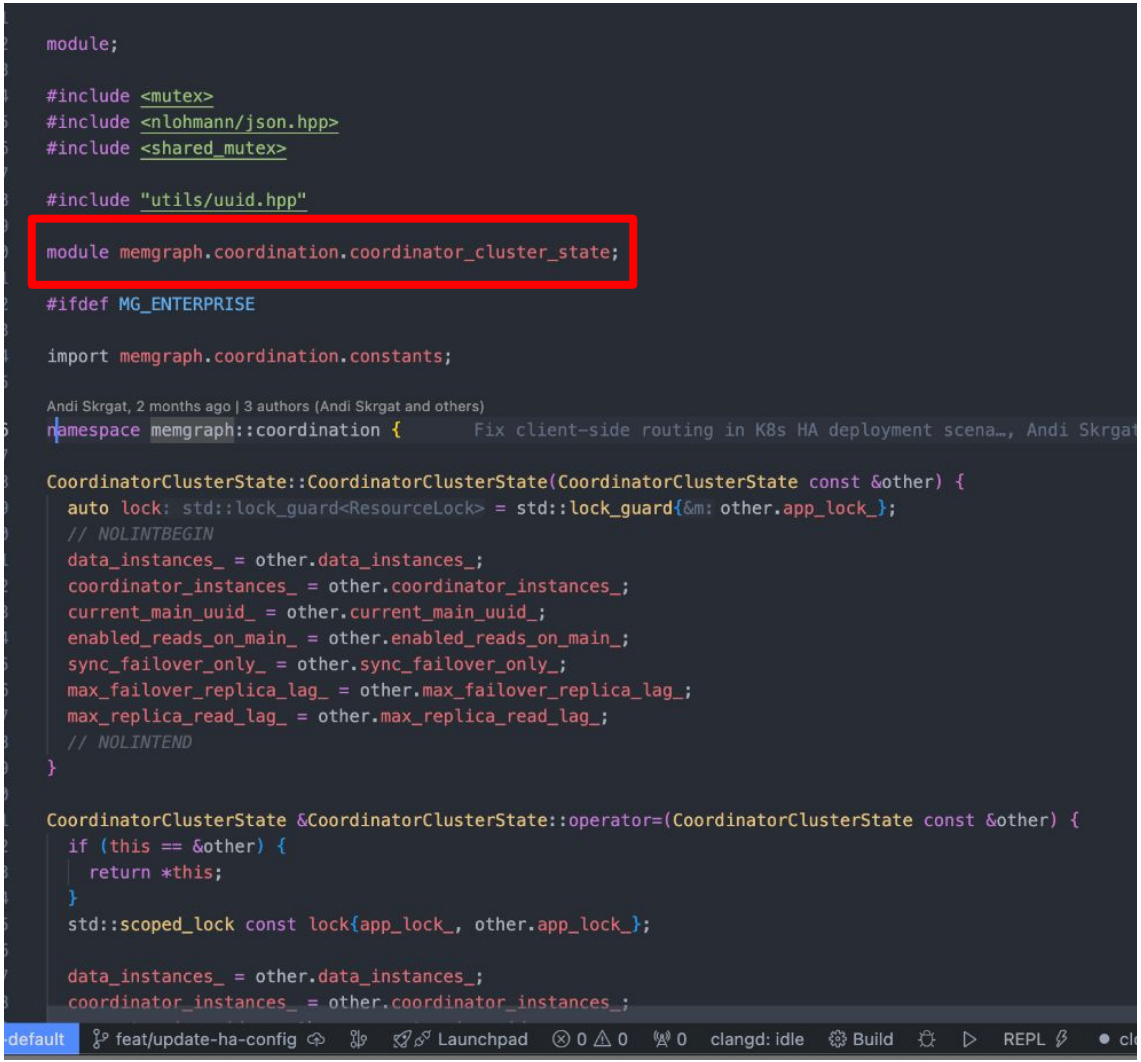
Key points

- For clang preferred extension is .cppm
- Modules are replacement for header files
- Global module fragment for includes
- Export means I am dealing with module interface units

coordination > coordinator_cluster_state.cppm > ...

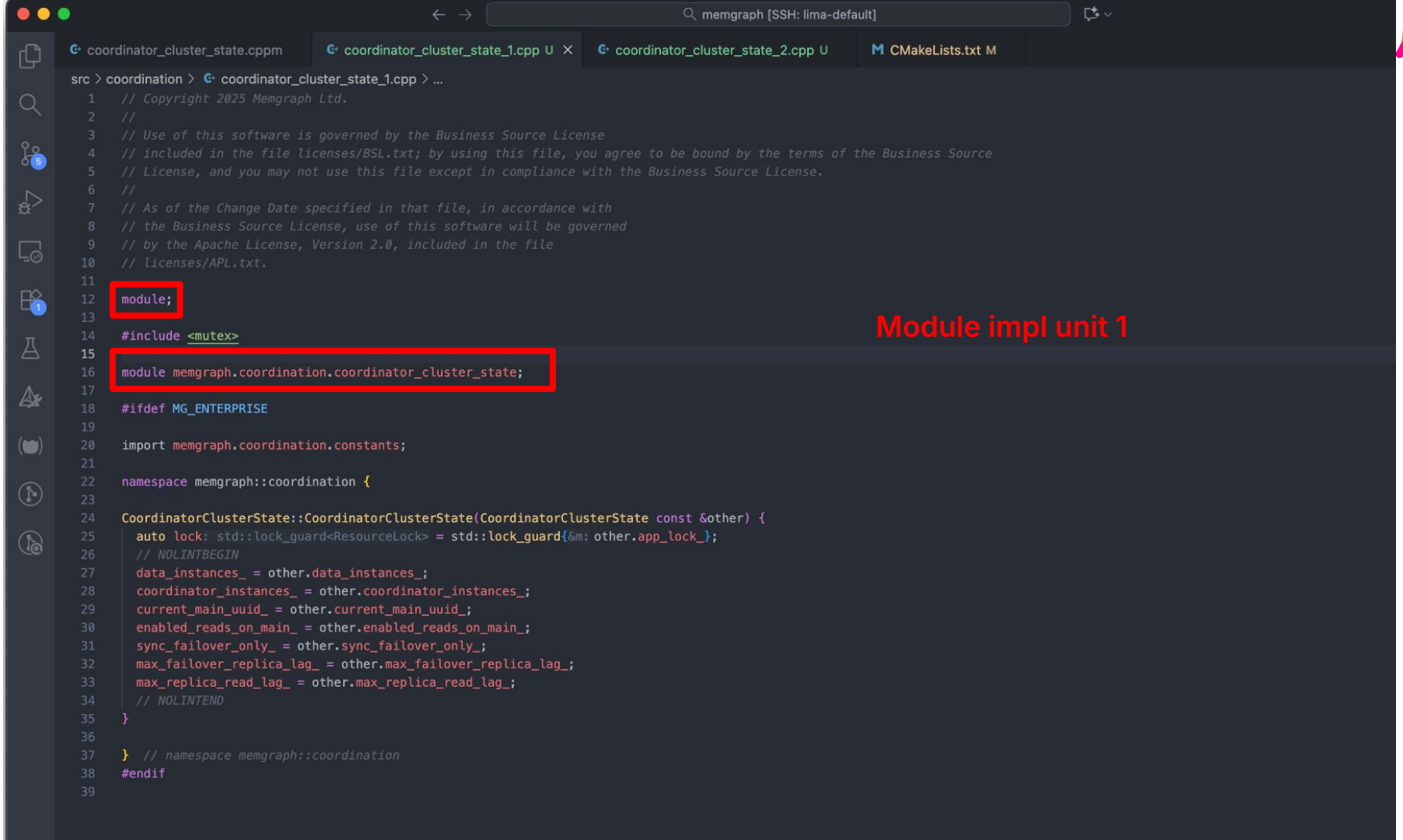
```
// Copyright 2025 Memgraph Ltd.
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.

module;

#include "replication_coordination_glue/role.hpp"
#include "utils/resource_lock.hpp"
#include "utils/uuid.hpp"

#include <libnuraft/nuraft.hxx>
#include <nlohmann/json_fwd.hpp>

#include <string>

export module memgraph.coordination.coordinator_cluster_state;

#ifdef MG_ENTERPRISE

import memgraph.coordination.coordinator_instance_context;      feat: Modularize cluster state, as51340 (18 hours ago)
import memgraph.coordination.data_instance_context;
import memgraph.coordination.coordinator_communication_config;

You, 18 hours ago | 4 authors (Andi Skrgat and others)
export namespace memgraph::coordination {

using nuraft::buffer;
// NOLINTNEXTLINE
using nuraft::buffer_serializer;
using nuraft::ptr;
// NOLINTNEXTLINE
using replication_coordination_glue::ReplicationRole;

Andi Skrgat, 2 months ago | 1 author (Andi Skrgat)
struct CoordinatorClusterStateDelta {
  std::optional<std::vector<DataInstanceContext>> data_instances_;
  std::optional<std::vector<CoordinatorInstanceContext>> coordinator_instances_;
  std::optional<utils::UUID> current_main_uuid_;
  std::optional<bool> enabled_reads_on_main_;
  std::optional<bool> sync_failover_only_;
  std::optional<uint64_t> max_failover_replica_lag_;
  std::optional<uint64_t> max_replica_read_lag_;

  bool operator==(const CoordinatorClusterStateDelta &other) const = default;
};
```


module;
#include <mutex>
#include <nlohmann/json.hpp>
#include <shared_mutex>
#include "utils/uuid.hpp"
module memgraph.coordination.coordinator_cluster_state;
#ifdef MG_ENTERPRISE
import memgraph.coordination.constants;
Andi Skrgat, 2 months ago | 3 authors (Andi Skrgat and others)
namespace memgraph::coordination {
Fix client-side routing in K8s HA deployment scena…, Andi Skrgat
CoordinatorClusterState::CoordinatorClusterState(CoordinatorClusterState const &other) {
auto lock: std::lock_guard<ResourceLock> = std::lock_guard{&m: other.app_lock_};
// NOLINTBEGIN
data_instances_ = other.data_instances_;
coordinator_instances_ = other.coordinator_instances_;
current_main_uuid_ = other.current_main_uuid_;
enabled_reads_on_main_ = other.enabled_reads_on_main_;
sync_failover_only_ = other.sync_failover_only_;
max_failover_replica_lag_ = other.max_failover_replica_lag_;
max_replica_read_lag_ = other.max_replica_read_lag_;
// NOLINTEND
}
CoordinatorClusterState &CoordinatorClusterState::operator=(CoordinatorClusterState const &other) {
if (this == &other) {
return *this;
}
std::scoped_lock const lock{app_lock_, other.app_lock_};
data_instances_ = other.data_instances_;
coordinator_instances_ = other.coordinator_instances_;
-default
feat/update-ha-config
Launchpad
0 0
0
clangd: idle
Build
REPL


module;
#include <mutex>
#include <nlohmann/json.hpp>
#include <shared_mutex>
#include "utils/uuid.hpp"
module memgraph.coordination.coordinator_cluster_state;
#ifdef MG_ENTERPRISE
import memgraph.coordination.constants;
Andi Skrgat, 2 months ago | 3 authors (Andi Skrgat and others)
namespace memgraph::coordination {
Fix client-side routing in K8s HA deployment scena…, Andi Skrgat
CoordinatorClusterState::CoordinatorClusterState(CoordinatorClusterState const &other) {
auto lock: std::lock_guard<ResourceLock> = std::lock_guard{&m: other.app_lock_};
// NOLINTBEGIN
data_instances_ = other.data_instances_;
coordinator_instances_ = other.coordinator_instances_;
current_main_uuid_ = other.current_main_uuid_;
enabled_reads_on_main_ = other.enabled_reads_on_main_;
sync_failover_only_ = other.sync_failover_only_;
max_failover_replica_lag_ = other.max_failover_replica_lag_;
max_replica_read_lag_ = other.max_replica_read_lag_;
// NOLINTEND
}
CoordinatorClusterState &CoordinatorClusterState::operator=(CoordinatorClusterState const &other) {
if (this == &other) {
return *this;
}
std::scoped_lock const lock{app_lock_, other.app_lock_};
data_instances_ = other.data_instances_;
coordinator_instances_ = other.coordinator_instances_;
default
feat/update-ha-config
Launchpad
0 0
0
clangd: idle
Build
REPL

Named modules

- A collection of module units with the same module name
- If there is an export ⇒ module interface units
- Everything else ⇒ module implementation units
- There can be more than one impl. units but only a single interface unit


memgraph [SSH: lima-default]
coordinator_cluster_state.cppm
coordinator_cluster_state_1.cpp U
coordinator_cluster_state_2.cpp U
CMakeLists.txt M
src > coordination > coordinator_cluster_state_1.cpp > ...
// Copyright 2025 Memgraph Ltd.
//
// Use of this software is governed by the Business Source License
// included in the file licenses/BSL.txt; by using this file, you agree to be bound by the terms of the Business Source
// License, and you may not use this file except in compliance with the Business Source License.
//
// As of the Change Date specified in that file, in accordance with
// the Business Source License, use of this software will be governed
// by the Apache License, Version 2.0, included in the file
// licenses/APL.txt.
module;
#include <mutex>
module memgraph.coordination.coordinator_cluster_state;
#ifdef MG_ENTERPRISE
import memgraph.coordination.constants;
namespace memgraph::coordination {
CoordinatorClusterState::CoordinatorClusterState(CoordinatorClusterState const &other) {
auto lock: std::lock_guard<ResourceLock> = std::lock_guard{&m: other.app_lock_};
// NOLINTBEGIN
data_instances_ = other.data_instances_;
coordinator_instances_ = other.coordinator_instances_;
current_main_uuid_ = other.current_main_uuid_;
enabled_reads_on_main_ = other.enabled_reads_on_main_;
sync_failover_only_ = other.sync_failover_only_;
max_failover_replica_lag_ = other.max_failover_replica_lag_;
max_replica_read_lag_ = other.max_replica_read_lag_;
// NOLINTEND
}
} // namespace memgraph::coordination
#endif
Module impl unit 1

memgraph [SSH: lima-default]

coordinator_cluster_state.cppm | coordinator_cluster_state_1.cpp U | coordinator_cluster_state_2.cpp U | CMakeLists.txt M

src > coordination > coordinator_cluster_state_2.cpp > ...

```
1  // Copyright 2025 Memgraph Ltd.
2  //
3  // Use of this software is governed by the Business Source License
4  // included in the file licenses/BSL.txt; by using this file, you agree to be bound by the terms of the Business Source
5  // License, and you may not use this file except in compliance with the Business Source License.
6  //
7  // As of the Change Date specified in that file, in accordance with
8  // the Business Source License, use of this software will be governed
9  // by the Apache License, Version 2.0, included in the file
10 // licenses/APL.txt.
11
12 module;
13
14 #include <mutex>
15 #include <nlohmann/json.hpp>
16 #include <shared_mutex>
17
18 #include "utils/uuid.hpp"
19
20 module memgraph.coordination.coordinator_cluster_state
21
22 #ifdef MG_ENTERPRISE
23
24 import memgraph.coordination.constants;
25
26 namespace memgraph::coordination {
27
28 CoordinatorClusterState &CoordinatorClusterState::operator=(CoordinatorClusterState const &other) {
29   if (this == &other) {
30     return *this;
31   }
32   std::scoped_lock const lock{app_lock_, other.app_lock_};
33
34   data_instances_ = other.data_instances_;
35   coordinator_instances_ = other.coordinator_instances_;
36   current_main_uuid_ = other.current_main_uuid_;
37   enabled_reads_on_main_ = other.enabled_reads_on_main_;
38   sync_failover_only_ = other.sync_failover_only_;
39   max_failover_replica_lag_ = other.max_failover_replica_lag_;
40   max_replica_read_lag_ = other.max_replica_read_lag_;
41   return *this;
42 }
43
44 CoordinatorClusterState::CoordinatorClusterState(CoordinatorClusterState &&other) noexcept
45     : data_instances_{std::move(other.data_instances_)},
46       coordinator_instances_{std::move(other.coordinator_instances_)},
```

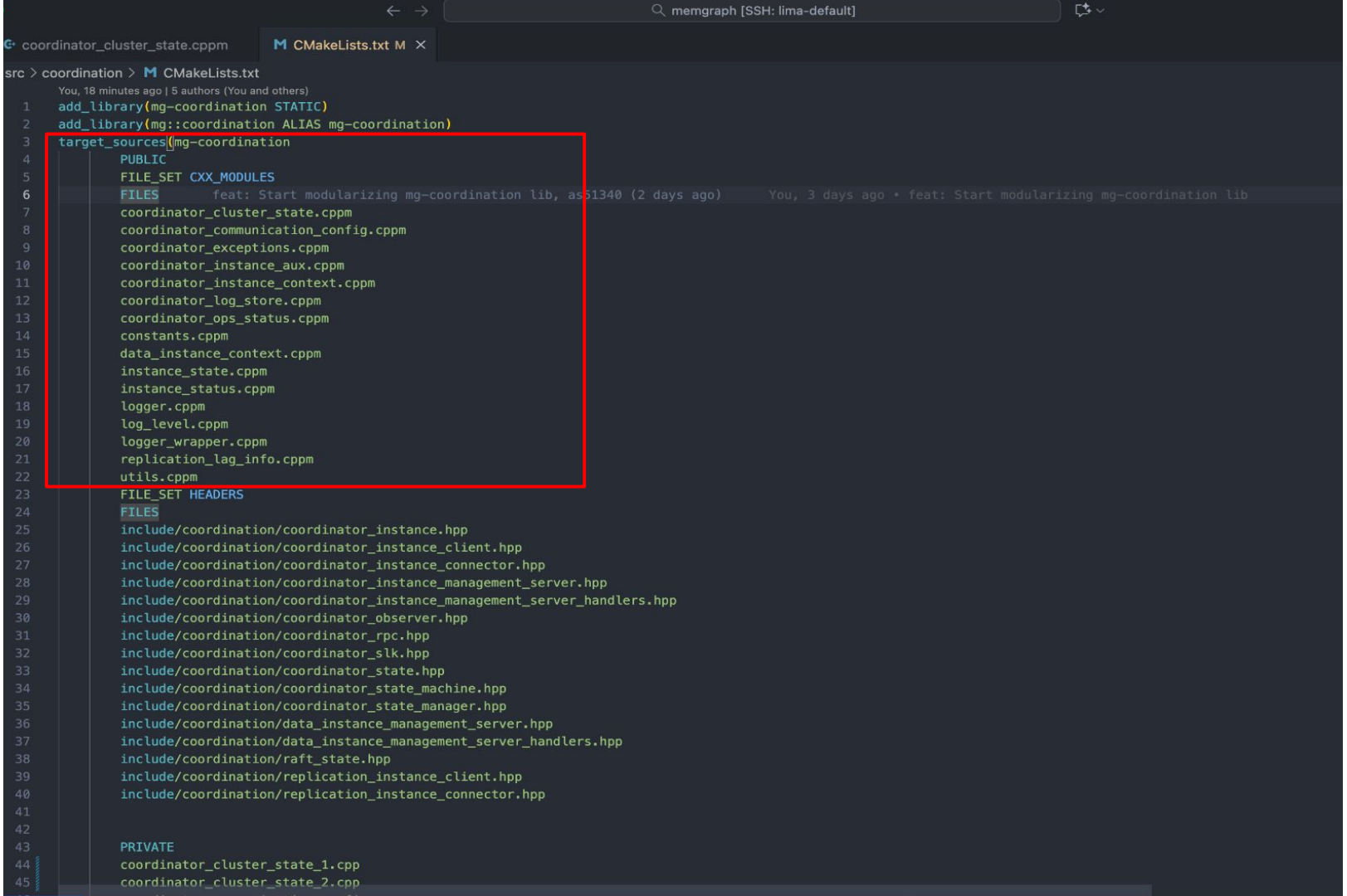
Module impl unit 2

lima-default | feat/update-ha-config* | Launchpad | 0 0 | 0 | clangd: idle | Build | REPL | clojure-lsp | Ln 17, Col 1 | Spaces: 2 | UTF-8

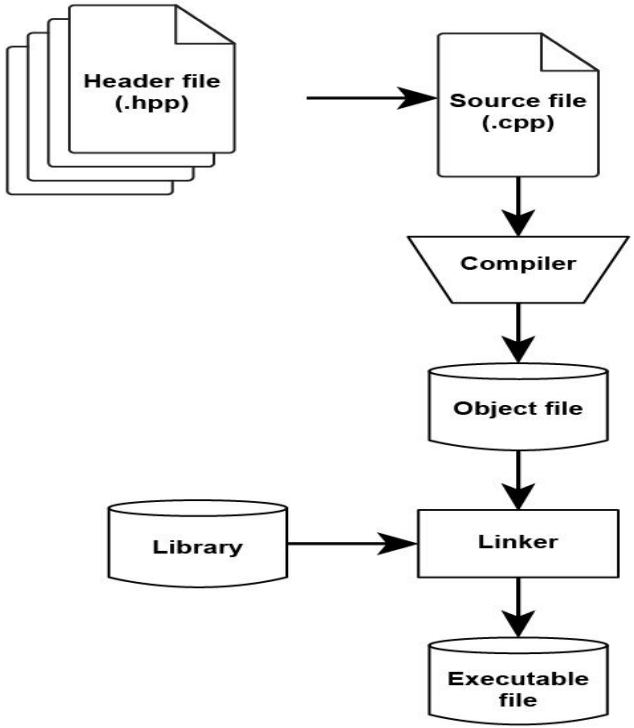
```
class CoordinatorInstanceManagementServer {
 public:
  explicit CoordinatorInstanceManagementServer(const ManagementServerConfig &config);
  CoordinatorInstanceManagementServer(const CoordinatorInstanceManagementServer &) = delete;
  CoordinatorInstanceManagementServer(CoordinatorInstanceManagementServer &&) = delete;
  CoordinatorInstanceManagementServer &operator=(const CoordinatorInstanceManagementServer &) = delete;
  CoordinatorInstanceManagementServer &operator=(CoordinatorInstanceManagementServer &&) = delete;

  ~CoordinatorInstanceManagementServer();

  bool Start();

  template <typename TRequestResponse, typename F>
  void Register(F &&callback) {
    rpc_server_.Register<TRequestResponse>(callback: std::forward<F>(callback));
  }

 private:
  communication::ServerContext rpc_server_context_;
  rpc::Server rpc_server_;
};
}  // namespace memgraph::coordination

module :private;
You, 22 hours ago | 1 author (You)
namespace memgraph::coordination {

You, 22 hours ago | 1 author (You)
namespace {

// NOTE: The coordinator server doesn't need more than 1 processing thread - it's not a bottleneck
constexpr auto kCoordInstanceManagementServerThreads: const int = 1;

}  // namespace

CoordinatorInstanceManagementServer::CoordinatorInstanceManagementServer(const ManagementServerConfig &config)
    : rpc_server_context_{communication::ServerContext{}},
      rpc_server_{endpoint: config.endpoint, context: &rpc_server_context_, workers_count: kCoordInstanceManagementServerThreads} {}

CoordinatorInstanceManagementServer::~CoordinatorInstanceManagementServer() {
  if (rpc_server_.IsRunning()) {
    rpc_server_.Shutdown();
  }
  rpc_server_.AwaitShutdown();
}

bool CoordinatorInstanceManagementServer::Start() { return rpc_server_.Start(); }

}  // namespace memgraph::coordination

#endif
```

Private module fragment

- Both declaration and implementation in the .cppm
- Implementation changes don’t require whole module recompilation and its dependencies
- You can have less files with the same functionality


memgraph [SSH: lima-default]
coordinator_cluster_state.cppm
CMakeLists.txt M
src > coordination > CMakeLists.txt
You, 18 minutes ago | 5 authors (You and others)
add_library(mg-coordination STATIC)
add_library(mg::coordination ALIAS mg-coordination)
target_sources(mg-coordination
PUBLIC
FILE_SET CXX_MODULES
FILES feat: Start modularizing mg-coordination lib, as61340 (2 days ago) You, 3 days ago • feat: Start modularizing mg-coordination lib
coordinator_cluster_state.cppm
coordinator_communication_config.cppm
coordinator_exceptions.cppm
coordinator_instance_aux.cppm
coordinator_instance_context.cppm
coordinator_log_store.cppm
coordinator_ops_status.cppm
constants.cppm
data_instance_context.cppm
instance_state.cppm
instance_status.cppm
logger.cppm
log_level.cppm
logger_wrapper.cppm
replication_lag_info.cppm
utils.cppm
FILE_SET HEADERS
FILES
include/coordination/coordinator_instance.hpp
include/coordination/coordinator_instance_client.hpp
include/coordination/coordinator_instance_connector.hpp
include/coordination/coordinator_instance_management_server.hpp
include/coordination/coordinator_instance_management_server_handlers.hpp
include/coordination/coordinator_observer.hpp
include/coordination/coordinator_rpc.hpp
include/coordination/coordinator_slk.hpp
include/coordination/coordinator_state.hpp
include/coordination/coordinator_state_machine.hpp
include/coordination/coordinator_state_manager.hpp
include/coordination/data_instance_management_server.hpp
include/coordination/data_instance_management_server_handlers.hpp
include/coordination/raft_state.hpp
include/coordination/replication_instance_client.hpp
include/coordination/replication_instance_connector.hpp
PRIVATE
coordinator_cluster_state_1.cpp
coordinator_cluster_state_2.cpp

Before modules



Standard translation unit

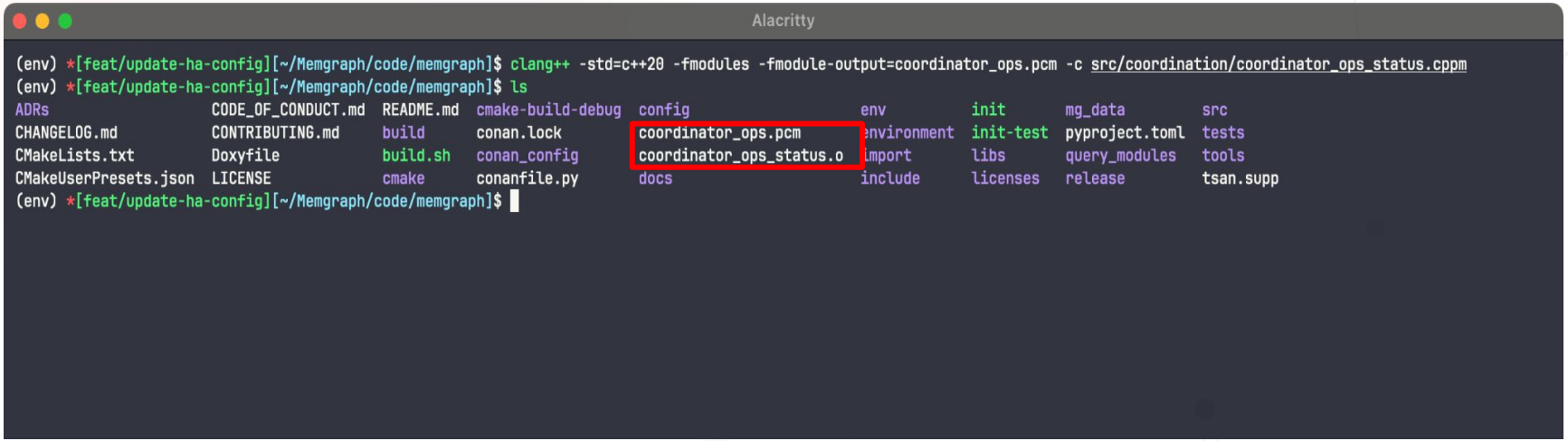
- Highly parallel build
- A lot of code gets propagated even if not used through transitive inclusion to the translation unit
- Relationship between a header file and translation unit is more a convention (cpp file can be used independently)

Modules

- Need an interface in order to import them
- When a module is compiled we get two files:
  - An object file for linking
  - Binary Module Interface (BMI)

Binary Module Inteface

- From OpenStd: The artifact created by a compiler to represent a module unit. The format is implementation specific.
  - All entities that are exported by a module in a format suitable for fast imports and lookups
- Reduced code bloating and better physical isolation but strong dependencies between modules ⇒ hard to parallelize, we are searching for DAG between modules
- clang-scan-deps tool used to find deps between modules


Alacritty
(env) *[feat/update-ha-config][~/Memgraph/code/memgraph]$ clang++ -std=c++20 -fmodules -fmodule-output=coordinator_ops.pcm -c src/coordination/coordinator_ops_status.cppm
(env) *[feat/update-ha-config][~/Memgraph/code/memgraph]$ ls
ADRs CODE_OF_CONDUCT.md README.md cmake-build-debug config env init mg_data src
CHANGELOG.md CONTRIBUTING.md build conan.lock coordinator_ops.pcm environment init-test pyproject.toml tests
CMakeLists.txt Doxyfile build.sh conan_config coordinator_ops_status.o import libs query_modules tools
CMakeUserPresets.json LICENSE cmake conanfile.py docs include licenses release tsan.supp
(env) *[feat/update-ha-config][~/Memgraph/code/memgraph]$

Modules != pre-compiled headers

- Pre-compiled headers still need to comply with the header inclusion model (linear dependency chain)
- Best to use PCH when having a big-sized header included at multiple places

Modularizing mg-coordination

- Experiment: Modularize mg-coordination and measure build times
  - Build time of memgraph
  - Build time when mg-coordination.a is missing
  - Size of mg-coordination.a
- https://github.com/memgraph/memgraph/pull/3491

Modules are space and time efficient

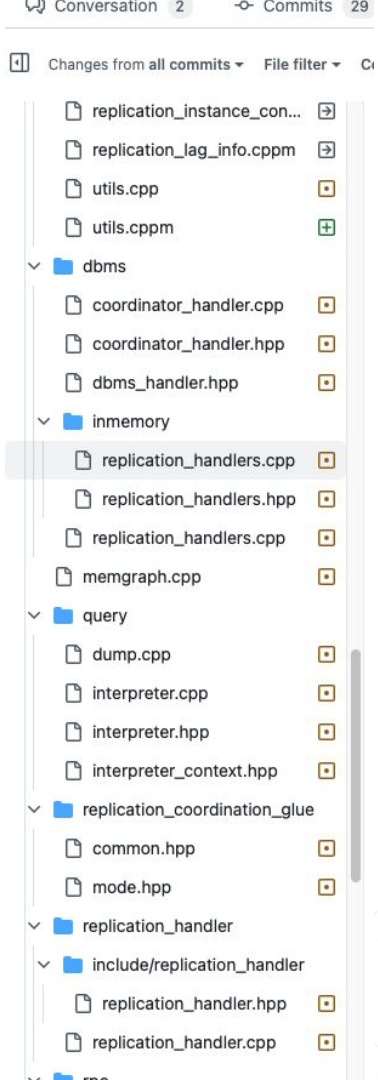
- Module contents aren’t transitively propagated if not requested explicitly
- For this you have **export import**
- Instead of including each header file into a cpp file and then compiling it, you are compiling it just once ⇒ smaller translation units
- You use BMI to only import some of the symbols from the module, not whole module code

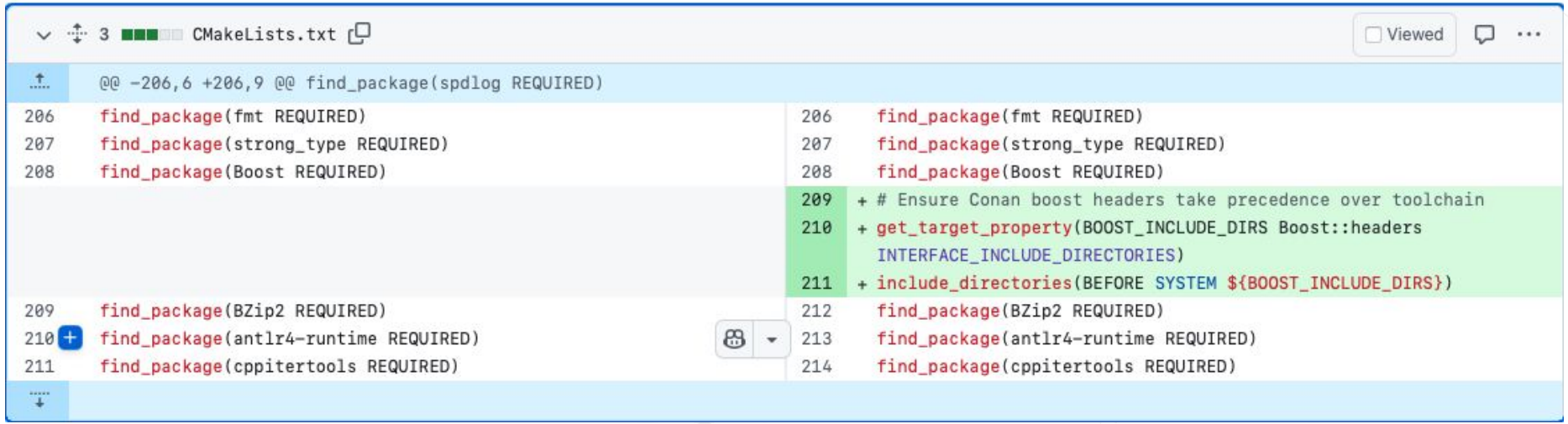
Results

	Build memgraph	Build libmg-coordination.a	Lib size
No modules	3:20.05	19.878	37MB
Modules	4:39.11	1:04.38	38MB

Modules lead to better abstractions

- Very fine-grained control over what's visible and what's not ⇒you can be very precise in specifying what are your needs
- Comparison with microservices
- Boost example with property_value.hpp


Conversation 2
Commits 29
Changes from all commits
File filter
replication_instance_con...
replication_lag_info.cppm
utils.cpp
utils.cppm
dbms
coordinator_handler.cpp
coordinator_handler.hpp
dbms_handler.hpp
inmemory
replication_handlers.cpp
replication_handlers.hpp
replication_handlers.cpp
memgraph.cpp
query
dump.cpp
interpreter.cpp
interpreter.hpp
interpreter_context.hpp
replication_coordination_glue
common.hpp
mode.hpp
replication_handler
include/replication_handler
replication_handler.hpp
replication_handler.cpp


3 CMakeLists.txt
Viewed
@@ -206,6 +206,9 @@ find_package(spdlog REQUIRED)
206 find_package(fmt REQUIRED)
207 find_package(strong_type REQUIRED)
208 find_package(Boost REQUIRED)
209 find_package(BZip2 REQUIRED)
210 find_package(antlr4-runtime REQUIRED)
211 find_package(cppitertools REQUIRED)
206 find_package(fmt REQUIRED)
207 find_package(strong_type REQUIRED)
208 find_package(Boost REQUIRED)
209 + # Ensure Conan boost headers take precedence over toolchain
210 + get_target_property(BOOST_INCLUDE_DIRS Boost::headers
INTERFACE_INCLUDE_DIRECTORIES)
211 + include_directories(BEFORE SYSTEM ${BOOST_INCLUDE_DIRS})
212 find_package(BZip2 REQUIRED)
213 find_package(antlr4-runtime REQUIRED)
214 find_package(cppitertools REQUIRED)

Caveats

- Not every hpp+cpp pair can be translated into a cppm with a private module fragment (example is the issue with nlohmann/json.hpp)
- Forward declarations don’t work across modulem, either use module partitions or refactor
- Hard to mix .hpp and .cppm files ⇒ couldn’t modularize “coordinator_rpc.hpp” independently from “rpc/utils.hpp”
- Build order matters, changes in the mg-coordination caused a failure to compile a property_value.hpp
- import std ⇒ not working in our case

Resources

- https://cmake.org/cmake/help/latest/command/target_sources.html#file-sets
- https://www.youtube.com/watch?v=7WK42YSfE9s&t=1s
- https://cryos.net/2024/01/c-modules-and-cmake/
- https://www.youtube.com/watch?v=_x9K9_q2ZXE&t=3209s
- https://www.youtube.com/watch?v=I_83IyxWGtE&t=3040s

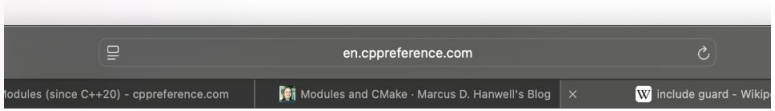
02

std::stacktrace

https://godbolt.org/z/jdPb75GMs

03

std::expected



Run this code

```
#include <cmath>
#include <expected>
#include <iomanip>
#include <iostream>
#include <string_view>

enum class parse_error
{
    invalid_input,
    overflow
};

auto parse_number(std::string_view& str) -> std::expected<double, parse_error>
{
    const char* begin = str.data();
    char* end;
    double retval = std::strtod(begin, &end);

    if (begin == end)
        return std::unexpected(parse_error::invalid_input);
    else if (std::isinf(retval))
        return std::unexpected(parse_error::overflow);

    str.remove_prefix(end - begin);
    return retval;
}

int main()
{
    auto process = [](std::string_view str)
    {
        std::cout << "str: " << std::quoted(str) << ", ";
        if (const auto num = parse_number(str); num.has_value())
            std::cout << "value: " << *num << '\n';
            // If num did not have a value, dereferencing num
            // would cause an undefined behavior, and
            // num.value() would throw std::bad_expected_access.
            // num.value_or(123) uses specified default value 123.
        else if (num.error() == parse_error::invalid_input)
            std::cout << "error: invalid input\n";
        else if (num.error() == parse_error::overflow)
            std::cout << "error: overflow\n";
        else
            std::cout << "unexpected!\n"; // or invoke std::unreachable();
    };

    for (auto src : {"42", "42abc", "meow", "inf"})
        process(src);
}
```

04

Practical

Task recommendations

- Figure out why modularizing mg-coordination slowed down the build ⇒ RESEARCH
- Modularize some other library (partially) and measure the effect ⇒ IMPL
- Make use of std::stacktrace in our exceptions
- Replace utils::BasicResult with std::expected
- Remove range-v3 dependency from Conan
- Std::move_only_function
-

Thank you for your time!

MEM GRAPH

www.memgraph.com