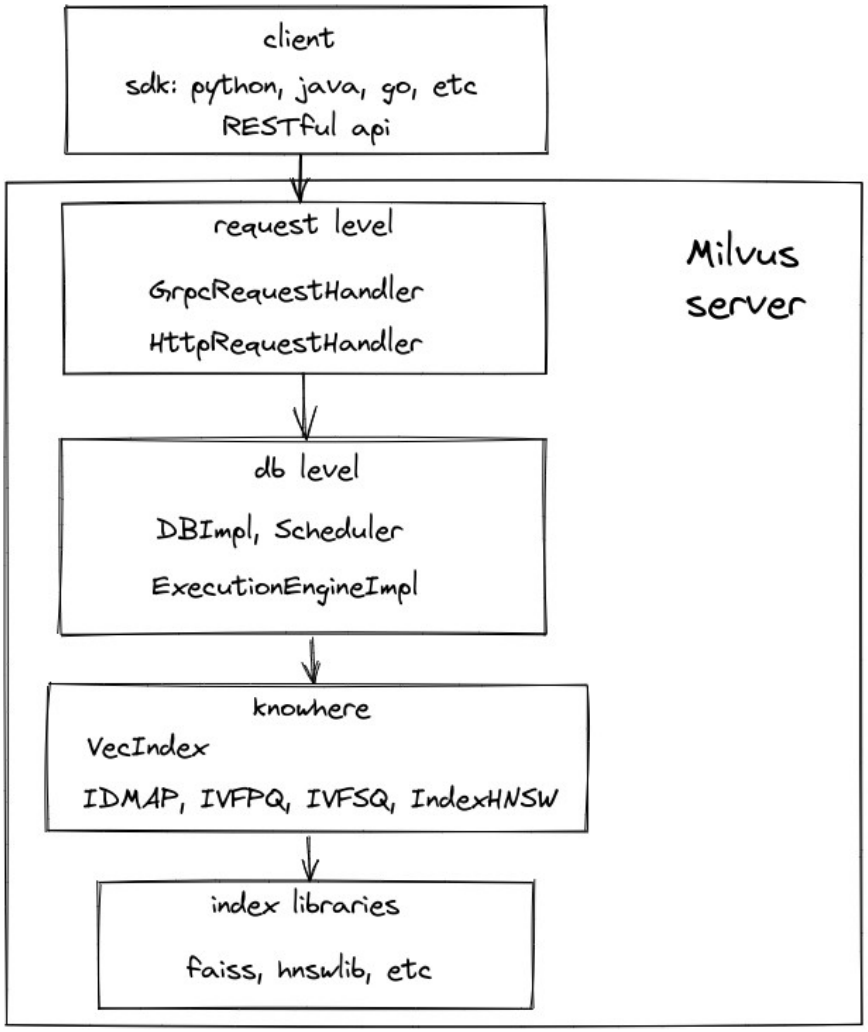
Milvus 1.0 call chain

Software Hierarchy

In the server side, we can split it to these level according to their roles and functions:

- request level: receive rpc/http requests, construct request pipeline
- db level: the core level, meta management, data management, task scheduler
- knowhere: a wrapper of different index libraries, provide unity interface
- index libraries: different index libraries including faiss, hnswlib



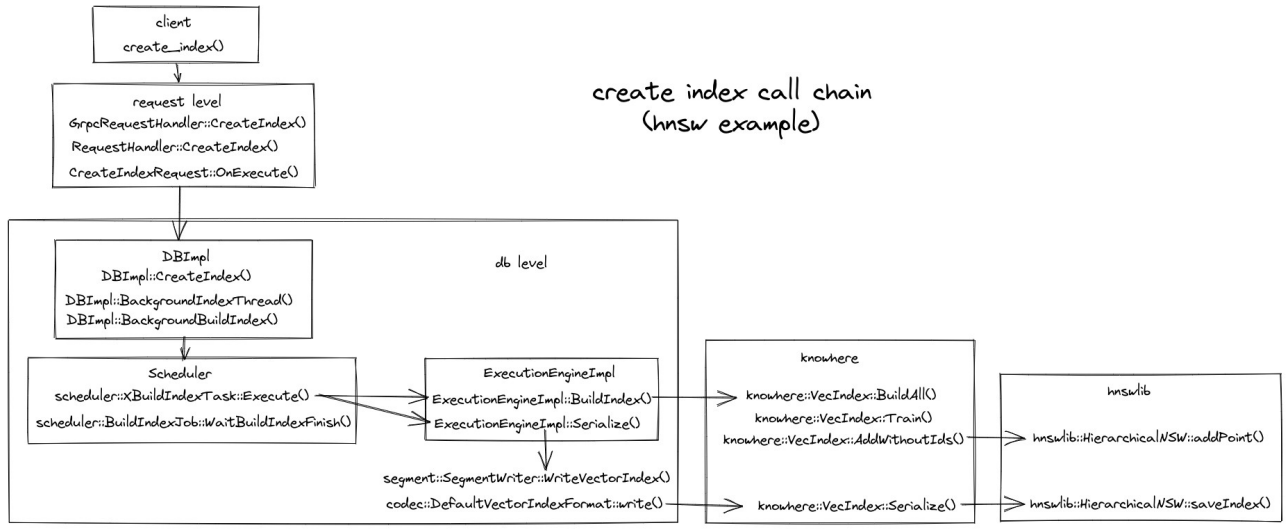
Related class files path

GrpcRequestHandler	src/server/grpc_impl/GrpcRequestHandler.cpp
RequestHandler	src/server/delivery/RequestHandler.cpp
CreateIndexRequest	src/server/delivery/request/CreateIndexRequest.cpp
PreloadCollectionRequest	src/server/delivery/request/PreloadCollectionRequest.cpp
SearchRequest	src/server/delivery/request/SearchRequest.cpp
DBImpl	src/db/DBImpl.cpp
scheduler::BuildIndexJob	src/scheduler/job/BuildIndexJob.cpp
scheduler::XbuildIndexTask	src/scheduler/task/BuildIndexTask.cpp

scheduler::SearchJob	src/scheduler/job/SearchJob.cpp
scheduler::XSearchTask	src/scheduler/task/XSearchTask.cpp
ExecutionEngineImpl	src/db/engine/ExecutionEngineImpl.cpp
segment::SegmentWriter	src/segment/SegmentWriter.cpp
segment::SegmentReader	src/segment/SegmentReader.cpp
codec::DefaultVectorIndexFormat	src/codecs/default/DefaultVectorIndexFormat.cpp
knowhere::VecIndex	src/index/knowhere/knowhere/index/vector_index/VecIndex.h
knowhere::VecIndexFactory	src/index/knowhere/knowhere/index/vector_index/VecIndexFactory.cpp
knowhere::IndexHNSW	src/index/knowhere/knowhere/index/vector_index/IndexHNSW.cpp
hnswlib::HierarchicalNSW	src/index/thirdparty/hnswlib/hnswalg.h

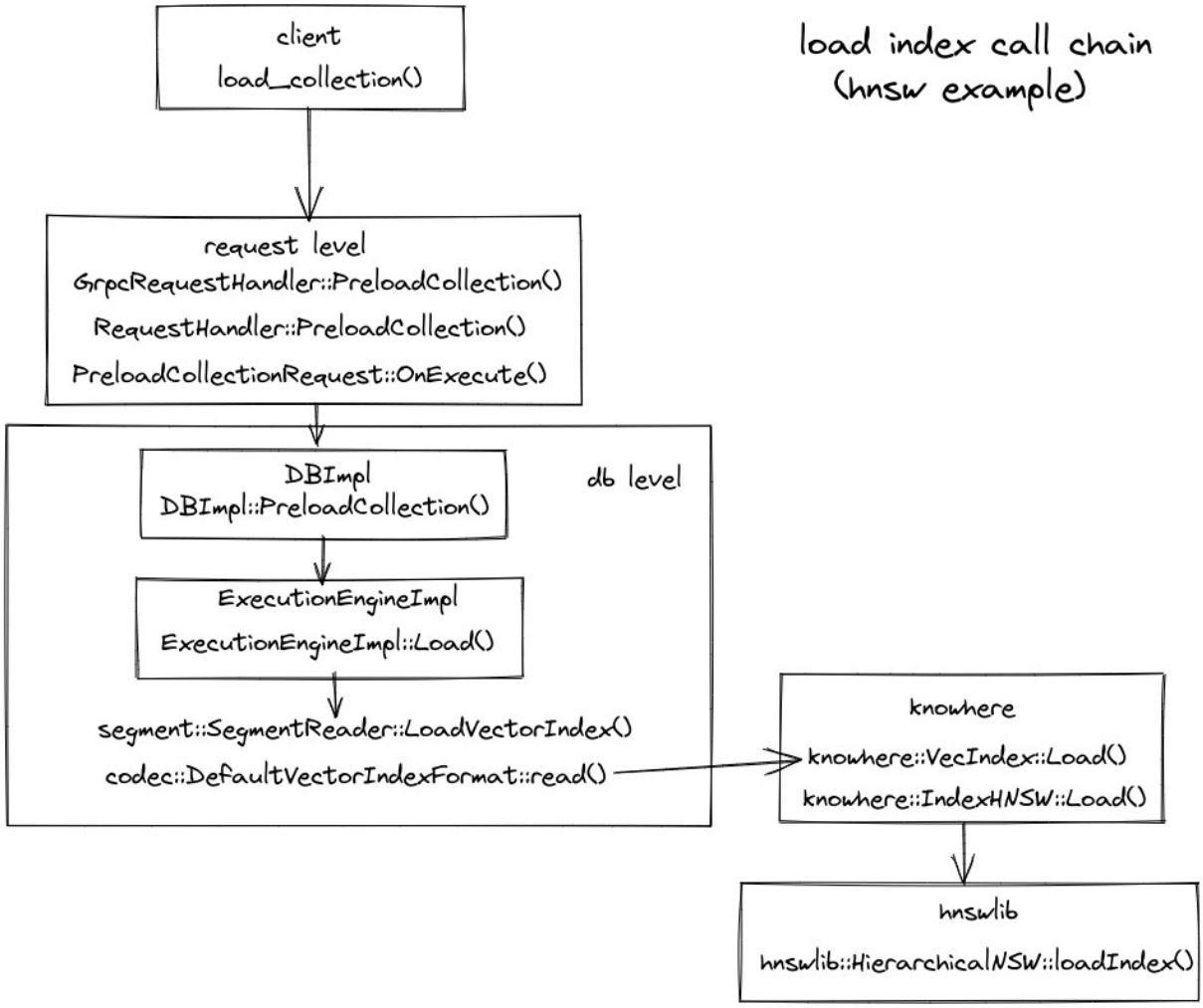
Call chain in main workflows

1. create index(hnsw example)

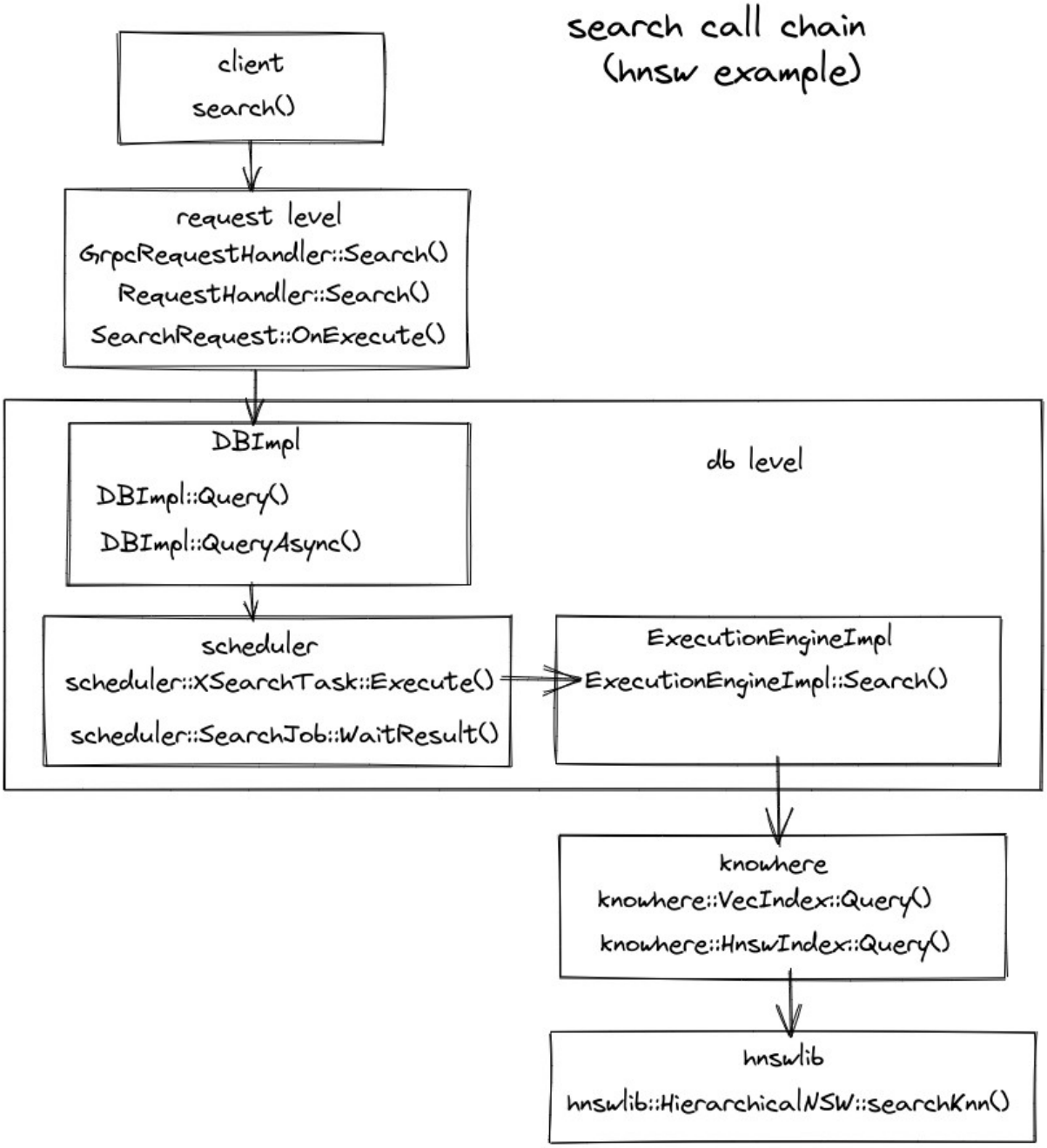


- When db level receive the create_index request, the DBImpl::CreateIndex() only update metadata and return. Then another background thread will be started by DBImpl::BackgroundIndexThread(), this work thread will build index for segments one by one.
- The DBImpl::BackgroundBuildIndex() will block at scheduler::BuildIndexJob::WaitBuildIndexFinish(), until scheduler::XBuildIndexTask::Execute() finished.
- ExecutionEngineImpl is a middle ware between knowere and milvus db level
- The knowhere::VecIndexFactory::CreateVecIndex() is to create available index instance for ExecutionEngineImpl, if we add a new index type, we must expand this function.
- segment::SegmentWriter is a class to persist index data to storage
- knowhere::VecIndex is an abstract class defines the unit interface for all index. (The raw data without index is defined as IDMAP, so we can treat it like other index type)
- Actually, the knowhere::VecIndex::Serialize() interface only return a binary data set, so that the caller can persist the data into storage(The codec::DefaultVectorIndexFormat::write() can do this).

2. load index(hnsw example)



3. search(hnsw example)


search call chain
(hnsw example)
client
search()
request level
GrpcRequestHandler::Search()
RequestHandler::Search()
SearchRequest::OnExecute()
DBImpl
DBImpl::Query()
DBImpl::QueryAsync()
db level
scheduler
scheduler::XSearchTask::Execute()
scheduler::SearchJob::WaitResult()
ExecutionEngineImpl
ExecutionEngineImpl::Search()
knowhere
knowhere::VecIndex::Query()
knowhere::HnswIndex::Query()
hnswlib
hnswlib::HierarchicalNSW::searchKnn()