

# Qdrant

## A Vector Search Engine in Rust



[github.com/qdrant/qdrant](https://github.com/qdrant/qdrant)

---

# Who am I

- Arnaud Gourlay
- Full time contributor @qdrant
- OSS and Rust <3
- [github.com/agourlay](https://github.com/agourlay)
- [agourlay.github.io](https://agourlay.github.io)



---

# Agenda

- About Qdrant (*pronounced quadrant*)
  - General introduction to vector search engines
  - Use cases for the technology
  - How it works internally
  - What makes Rust a good fit for the job
-

---

# What is Qdrant?

- Open source vector search engine
- Written in Rust
- Interactions via HTTP/JSON or gRPC
- Official clients in Python, Rust & Go
- Distributed deployment
- SaaS cloud offering

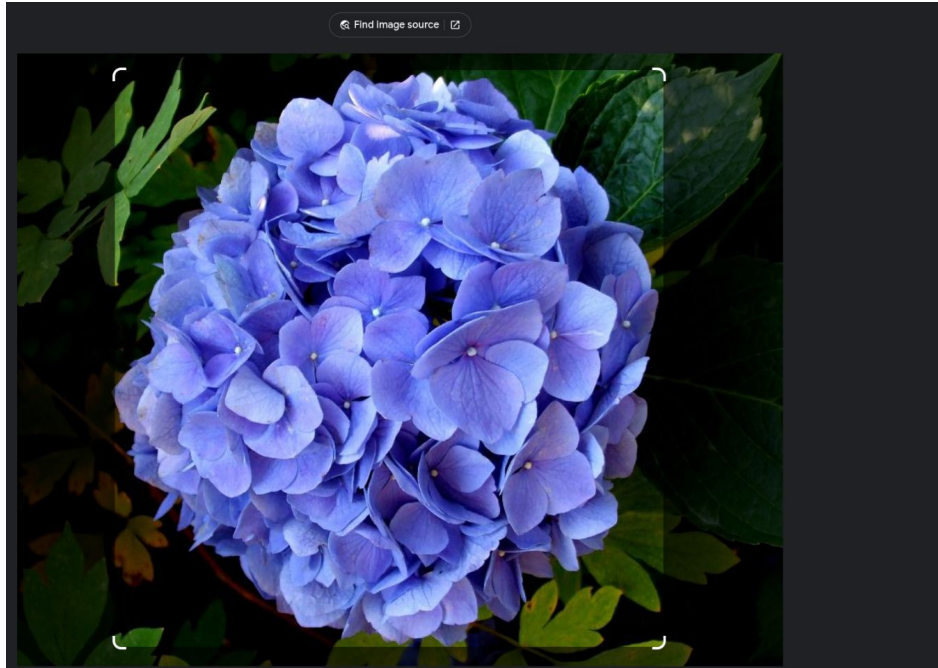


---

# Evolution of search

- Traditionally exact match on a text keyword
  - Growing amount of unstructured data
  - Recommendations are everywhere
  - Similarity search
-

# Reverse image search (Google Lens)



French hydrangea  
Plant

Search

Visual matches

- eastcoastrseries... Bigleaf Hydrangea Hydrangea macrophyll...
- pinterest.com Hydrangea my favorite flower! Love all the...
- installindirect.com Tips for Growing Hydrangeas in Souther...
- missouri.edu Hydrangeas, the bold chamaleon of plants [...]
- lesleinsley.com "What Makes The Hydrangeas So Blue?" ...
- flowerpower.com.au Hydrangea Blue
- inquirer.com Why your hydrangeas may go wild this year
- grasspad.com Best Hydrangeas for Shade and Sun - Grass...
- covingtonnursery.co... LYNDANCEA n... bruns.de Hydrangea macrophylla
- pinterest.com #hydrangea the #flower of the #hamptons [...]
- pinterest.com What Color is Periwinkle Blue | Periwinkle Blue...

---

# Neural search

- Real example on (Simple) Wikipedia (170k documents)
- Query: `What is the capital of the United States?`
- Top-3 Hits

## Lexical Search (BM25)

- **Capital** punishment (the death penalty) has existed in the **United States** [...]
- Ohio is one of the 50 **states** in the **United States**. Its **capital** is Columbus. [...]
- Nevada is one of the **United States'** **states**. Its **capital** [...]

## Neural Search

- Washington, D.C. [...] is the **capital of the United States**. [...]
- A capital city (or capital town or just capital) is a city or town, [...]
- The United States **Capitol** is the building where the United States Congress meets [...]

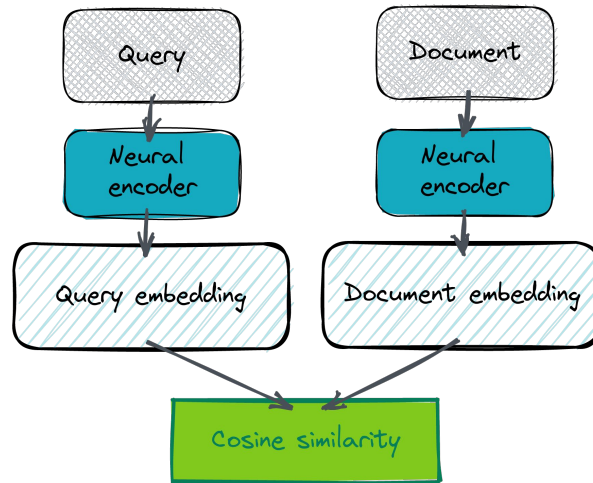
source: Nils Reimer

---

---

# Similarity models

- Trained via machine learning
- “similar” inputs have vectors “close to each other” in space





---

# Vector search engine

- Enables similarity search
  - Storage: persists durably vector embeddings
  - Search: find vectors most similar to an input vector
  - Efficiently!
-

---

# Usage example

Python

```
1 from sentence_transformers import SentenceTransformer
2 from tqdm import tqdm
3 tqdm.pandas()
4
5 # load the model
6 model = SentenceTransformer('multi-qa-MiniLM-L6-cos-v1')
7
8 # encode
9 df["ve
```

Python

```
1 from qdrant_client import QdrantClient
2 from qdrant_client.http.models import * #VectorParams
3
4 client = QdrantClient
5 client.recreate_collection(
6     collection_name=
7     vectors_config=V
8 )
```

Python

```
1 search_term = "earth observation"
2
3 search_result = client.search(
4     collection_name="test_collection",
5     query_vector=model.encode(search_term),
6     limit=3
7 )
8
9 search_result
```

Source: <https://geo.rocks/post/geospatial-vector-search-qdrant/>

---

---

# Vector payload

- Attach additional data to a vector
- Filtered search on payload fields
- Text keyword, numeric, geo coordinates ...

```
from qdrant_client import QdrantClient
from qdrant_client.http import models

client = QdrantClient(host="localhost", port=6333)

client.upsert(
    collection_name="{collection_name}",
    points=[
        models.PointStruct(
            id=1,
            vector=[0.05, 0.61, 0.76, 0.74],
            payload={
                "city": "Berlin",
                "price": 1.99,
            },
        ),
        models.PointStruct(
            id=2,
            vector=[0.19, 0.81, 0.75, 0.11],
            payload={
                "city": ["Berlin", "London"],
                "price": 1.99,
            },
        ),
        models.PointStruct(
            id=3,
            vector=[0.36, 0.55, 0.47, 0.94],
            payload={
                "city": ["Berlin", "Moscow"],
                "price": [1.99, 2.99],
            },
        ),
    ],
)
```

---

# Naive vector search

- Store vectors in a “table” (4 dimensions in example)
- Compute similarity between input vector and **each** vector
- Return vector id with max similarity

id	w	x	y	z
1	1.22	24.61	8.79	49.08
2	3.45	13.09	44.32	2.27
3	0.05	67.54	76.87	6.91
...	...	...	...	...

---

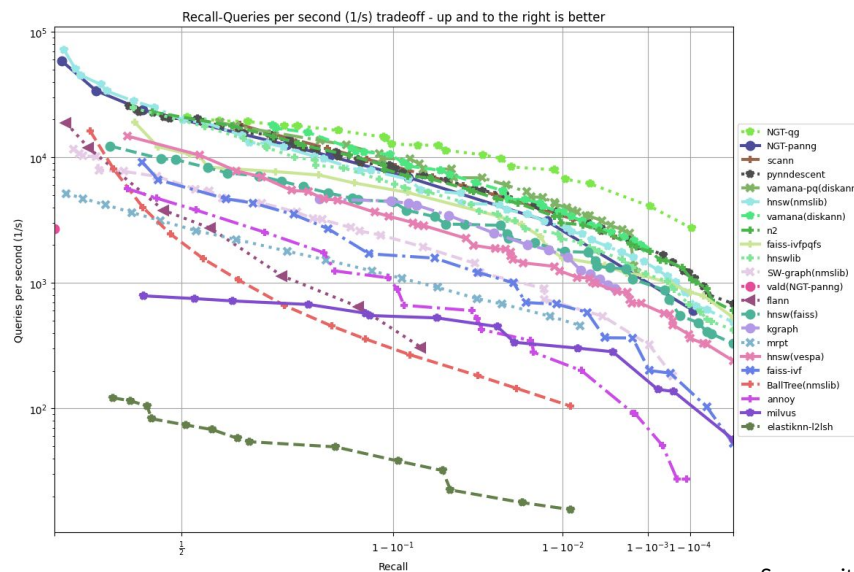
---

# We need a vector index

- Traditional indexes do not fit
  - Geospatial indexes (KD-trees)
  - k-nearest neighbors (kNN)
  - Curse of dimensionality
  - How to handle very large dimensions?
-

# ANN Search

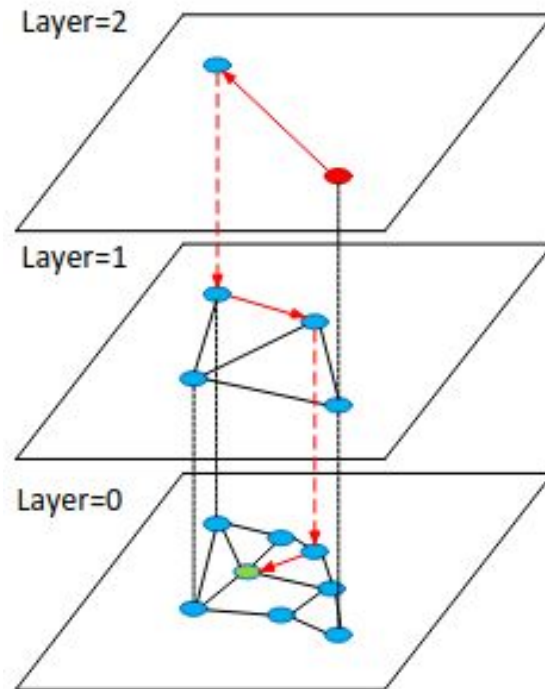
- Approximate Nearest Neighbors
- Tradeoff: precision vs speed



---

# Hierarchical Navigable Small Worlds

- Proximity graphs
- Skip-List
- 'M' friends per vector
- 'efSearch' per layer



\*Source: [HNSW paper](#)

---

---

# HNSW filtering

- Custom implementation to support payload filters
  - Post-filtering vs Pre-filtering
  - Enrich graph with indexed payload info
  - Single stage search
  - <https://qdrant.tech/articles/filterable-hnsw/>
-



---

# Distance metrics

- Very common operation (indexing & search)
- Several similarity metrics available depending on encoder

```
pub trait Metric {  
    /// Enum value  
    fn distance() -> Distance;  
    /// Greater the value - closer the vectors  
    fn similarity(v1: &[f32], v2: &[f32]) -> f32;  
    /// Necessary vector transformations performed before adding it to the collection (normalization)  
    /// Return None if metric does not required preprocessing  
    fn preprocess(vector: &[f32]) -> Option<Vec<f32>>;  
}
```

---

---

# Naive Dot product

```
impl Metric for DotProductMetric {  
    fn distance() -> Distance {  
        Distance::Dot  
    }  
  
    fn similarity(v1: &[f32], v2: &[f32]) -> ScoreType {  
        v1.iter().zip(v2).map(|(a, b)| a * b).sum()  
    }  
}
```

---

# Dot product performance

- 70% CPU in “core..iter..traits..accum..Sum\$GT\$::sum::”

```
SLTf32u20$asu20$score..iter..traits..accum..Sum$GT$::sum::: $u7b5$u7b$closure$u7d$u7d$: :h699d7e6e9e9a33e
core::iter::adapters::map::map_fold::: $u7b5$u7b$closure$u7d$u7d$: :hd2d1a5e6ee2934c0
core::iter::traits::iterator::Iterator::fold::: h112598803a0051a0
SLTScore..iter..adapters::map::MapSLTf32u20$asu20$score..iter..traits..iterator::Iterator$GTS::fold::: h2218f6b789d35668
SLTf32u20$asu20$score..iter..traits..accum..Sum$GT$::sum::: hfcf98caaff52283e
core::iter::traits::iterator::Iterator::sum::: h4b0d4043e3d7b1
segment::spaces::simple::dot_similarity::: hebffc0d5e28f1
SLT$segment..spaces..simple..DotProductMetrics$u20$asu20$segment..spaces..metric..Metric$GTS::similarity::: h814284886452596
SLT$segment..vector_storage..simple_vector_storage..SimpleVectorStorageSLT$Metric$GTS$u20$asu20$segment..vector_storage..vector_storage_base..VectorStorage$GTS::score_all::: $u7b5$u7b$closure$u7d$u7d$: :h8103d823c76bc3
core::ops::function::impls::: SLT$impl$u20$score..ops..function..FnOnceSLT$AS$GTS$u20$for$u20$SRF$mut$u20$SF$GTS::call_once::: h45494437a593efc
core::option::OptionSLT$T$GTS::map::: h9a06f012f61ada30
SLTScore..iter..adapters::map::MapSLTf32u20$asu20$score..iter..traits..iterator::Iterator$GTS::next::: h4b462cf4d9b73222
segment::spaces::tools::peek_top_largest_iterable::: h7b0b6a6dec9a0d5
SLT$segment..vector_storage..simple_vector_storage..SimpleVectorStorageSLT$Metric$GTS$u20$asu20$segment..vector_storage..vector_storage_base..VectorStorage$GTS::score_all::: h35b722a7618011ed
segment::index::hnsw_index::point_scorer::FilteredScorer::score_points::: h2e1661ca118eabb9
segment::index::hnsw_index::graph_layers::GraphLayersBase::search_on_level::: h6cab1c32f7572a0
segment::index::hnsw_index::graph_layers::GraphLayersBase::search_on_level::: h507c991569bab4e6
segment::index::hnsw_index::graph_layers::GraphLayersSLT$T$GraphLinks$GTS::search::: h53baf9d997a62bf
segment::index::hnsw_index::hnsw::HNSWIndexSLT$T$GraphLinks$GTS::search_with_graph::: h694d6ef1cc6cd6d
segment::index::hnsw_index::hnsw::HNSWIndexSLT$T$GraphLinks$GTS::search_vectors_with_graph::: $u7b5$u7b$closure$u7d$u7d$: :hd556372b68852f8
core::iter::adapters::map::map_fold::: $u7b5$u7b$closure$u7d$u7d$: :hd2d1a5e6ee2934c0
core::iter::traits::iterator::Iterator::fold::: h112598803a0051a0
SLTScore..iter..adapters::map::MapSLTf32u20$asu20$score..iter..traits..iterator::Iterator$GTS::fold::: h2218f6b789d35668
SLTf32u20$asu20$score..iter..traits..accum..Sum$GT$::sum::: hfcf98caaff52283e
core::iter::traits::iterator::Iterator::sum::: h4b0d4043e3d7b1
segment::spaces::simple::dot_similarity::: hebffc0d5e28f1
SLT$segment..spaces..simple..DotProductMetrics$u20$asu20$segment..spaces..metric..Metric$GTS::similarity::: h814284886452596
SLT$segment..vector_storage..simple_vector_storage..SimpleVectorStorageSLT$Metric$GTS$u20$asu20$segment..vector_storage..vector_storage_base..VectorStorage$GTS::score_all::: $u7b5$u7b$closure$u7d$u7d$: :h8103d823c76bc3
core::ops::function::impls::: SLT$impl$u20$score..ops..function..FnOnceSLT$AS$GTS$u20$for$u20$SRF$mut$u20$SF$GTS::call_once::: h45494437a593efc
core::option::OptionSLT$T$GTS::map::: h9a06f012f61ada30
SLTScore..iter..adapters::map::MapSLTf32u20$asu20$score..iter..traits..iterator::Iterator$GTS::next::: h4b462cf4d9b73222
segment::spaces::tools::peek_top_largest_iterable::: h7b0b6a6dec9a0d5
SLT$segment..vector_storage..simple_vector_storage..SimpleVectorStorageSLT$Metric$GTS$u20$asu20$segment..vector_storage..vector_storage_base..VectorStorage$GTS::score_all::: h35b722a7618011ed
SLT$LaLoc_vec..VecSLT$T$GTS$u20$asu20$score..iter..traits..collect..FromIteratorSLT$T$GTS$GTS::from_iter::: h1c28c5b7692b66e
core::iter::traits::iterator::Iterator::collect::: h71b2d61495e506
segment::index::hnsw_index::hnsw::HNSWIndexSLT$T$GraphLinks$GTS::search_vectors_with_graph::: hae4ce8aeb8414
SLT$segment..index..hnsw_index..hnsw..HNSWIndexSLT$T$GraphLinks$GTS$u20$asu20$segment..index..index_base..VectorIndex$GTS::search_batch::: h47d0f94e78dbadb
SLT$segment..segment..Segment$u20$asu20$segment..entry..entry_point..SegmentEntry$GTS::search_batch::: h4555ba9fe70f9e
collection::collection_manager::segments::searcher::search_in_segment::: $u7b5$u7b$closure$u7d$u7d$: :h665e57b02e808e7
tokio::runtime::task::core::core$SLT$K$S$GTS::poll::: $u7b5$u7b$closure$u7d$u7d$: :h8dce5f271a8f949
tokio::loom::std::uname_cell::UnsafeCellSLT$T$GTS::with_mut::: h3d4766a17374151e
tokio::runtime::task::raw::RawTask::poll::: h385014b6f18066a
tokio::runtime::task::LocalNotifiedSLT$S$GTS::run::: h5acc35bc1e0f13e1
tokio::runtime::scheduler::multi_thread::worker::Context::run_task::: $u7b5$u7b$closure$u7d$u7d$: :h43d9b3ca5b8ca12d
```

---

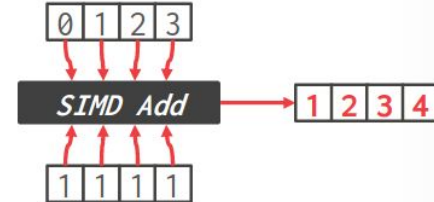
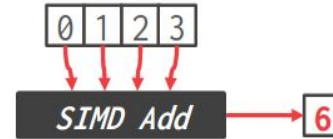
# SIMD

- Single Instruction Multiple Data.
  - Same operation on multiple data simultaneously!
  - X86: SSEs, AVXs
  - ARM: NEON
  - Hand-coded or auto-vectorizing compiler.
-

---

# Horizontal vs vertical

- Horizontal
- Vertical



\*source: CMU CS Andy Pavlo

---

---

# SIMD dynamic feature detections

```
fn similarity(v1: &[VectorElementType], v2: &[VectorElementType]) -> ScoreType {
    #[cfg(target_arch = "x86_64")]
    {
        if is_x86_feature_detected!("avx")
            && is_x86_feature_detected!("fma")
            && v1.len() >= MIN_DIM_SIZE_AVX
        {
            return unsafe { dot_similarity_avx(v1, v2) };
        }
    }

    #[cfg(any(target_arch = "x86", target_arch = "x86_64"))]
    {
        if is_x86_feature_detected!("sse") && v1.len() >= MIN_DIM_SIZE_SIMD {
            return unsafe { dot_similarity_sse(v1, v2) };
        }
    }

    #[cfg(all(target_arch = "aarch64", target_feature = "neon"))]
    {
        if std::arch::is_aarch64_feature_detected!("neon") && v1.len() >= MIN_DIM_SIZE_SIMD {
            return unsafe { dot_similarity_neon(v1, v2) };
        }
    }

    dot_similarity(v1, v2)
}
```

---

---

# AVX SIMD dot product

```
use std::arch::x86_64::*;

#[target_feature(enable = "avx")]
#[target_feature(enable = "fma")]
pub unsafe fn dot_avx(v1: &[f32], v2: &[f32]) -> f32 {
    let mut sum256: __m256 = _mm256_setzero_ps(); // accumulated sum
    for i : usize in 0..v1.len() / 8 {
        let v1_256: __m256 = _mm256_loadu_ps(mem_addr: v1[8 * i..].as_ptr()); // vector floats from v1
        let v2_256: __m256 = _mm256_loadu_ps(mem_addr: v2[8 * i..].as_ptr()); // vector floats from v2
        sum256 = _mm256_fmadd_ps(a: v1_256, b: v2_256, c: sum256); // sum256 += v1_256 * v2_256
    }

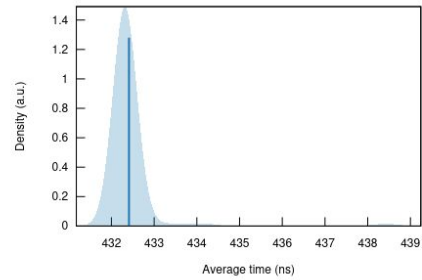
    // let's sum all elements in sum256
    let sum128: __m128 = _mm_add_ps(a: _mm256_extractf128_ps(a: sum256, 1), b: _mm256_castps256_ps128(a: sum256));
    let sum64: __m128 = _mm_add_ps(a: sum128, b: _mm_movehl_ps(a: sum128, b: sum128));
    let sum32: __m128 = _mm_add_ss(a: sum64, b: _mm_shuffle_ps(a: sum64, b: sum64, 0x55));
    _mm_cvtss_f32(a: sum32)
}
```

---

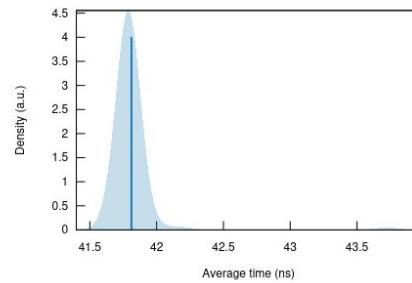
---

# Criterion benchmark

dot-product-group/dot\_iter



dot-product-group/dot\_simd





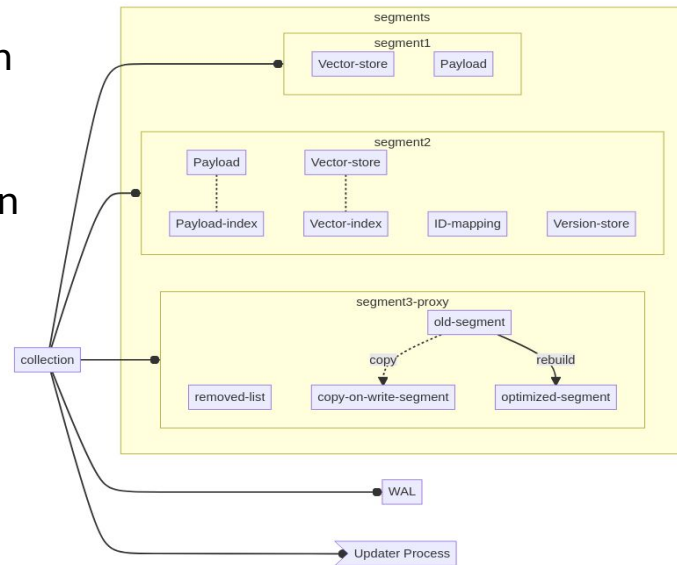
---

# SIMD indexing impact

- 100k vectors of dim. 500
  - HNSW index with dot product
  - Dot iterator: 333 seconds
  - Dot SIMD: 95 seconds
-

# Vector collection

- Several segments per collection
- Persistence with RocksDB
- Optimizers keeping things clean
- In memory vs memmap



---

# Concurrent programming

- Qdrant is inherently stateful
  - Manage concurrent accesses
  - Threads sharing data via Channel
  - Threads synchronizing on Mutex/RwLock
-

---

# RwLock

- API enforces proper usage
- No unlock method!
- Read guard impl. Deref
- Write guard impl. DerefMut
- Check out parking\_lot

```
use std::sync::RwLock;

let lock = RwLock::new(5);

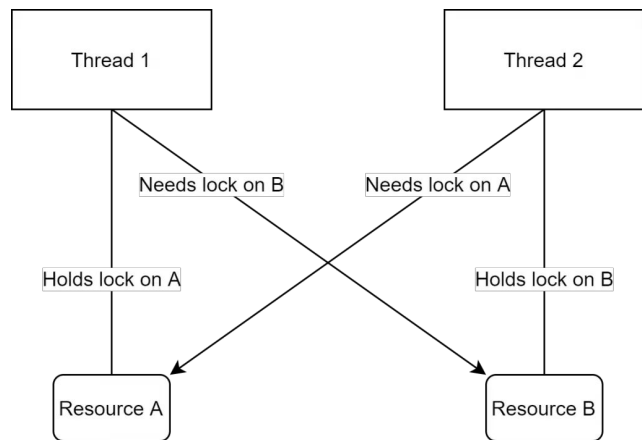
// many reader locks can be held at once
{
    let r1 = lock.read().unwrap();
    let r2 = lock.read().unwrap();
    assert_eq!(*r1, 5);
    assert_eq!(*r2, 5);
} // read locks are dropped at this point

// only one write lock may be held, however
{
    let mut w = lock.write().unwrap();
    *w += 1;
    assert_eq!(*w, 6);
} // write lock is dropped here
```

---

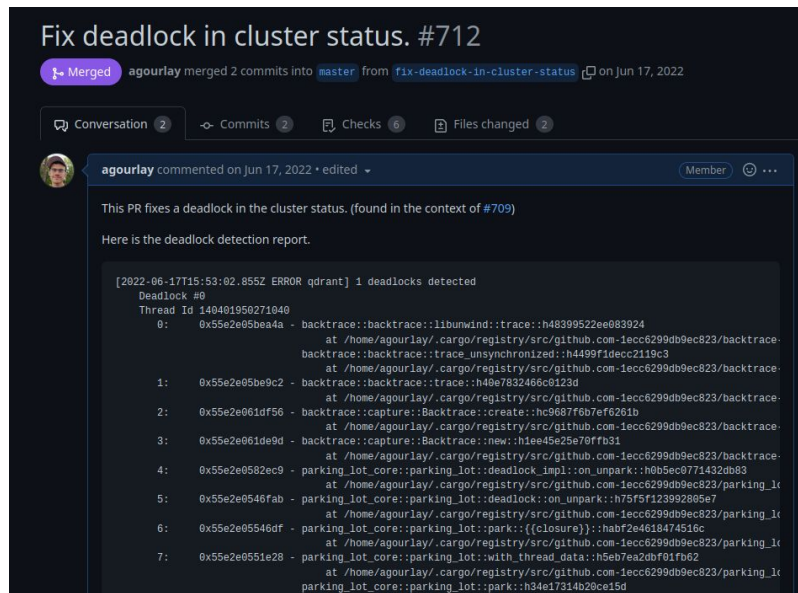
# Dead locks

- Threads waiting for each other
- Double reads on the same thread
- Not only locks
- Not caught by rustc
- Requires discipline



# Runtime deadlock detector

- parking\_lot “deadlock\_detection” build feature



Fix deadlock in cluster status. #712

Merged agourlay merged 2 commits into master from fix-deadlock-in-cluster-status on Jun 17, 2022

agourlay commented on Jun 17, 2022 • edited

This PR fixes a deadlock in the cluster status. (found in the context of #709)


Here is the deadlock detection report.



```
[2022-06-17T15:53:02.855Z ERROR qdrant] 1 deadlocks detected
Deadlock #0
Thread Id 140401950271040
0: 0x55e2e05bea4a - backtrace::backtrace::libunwind::trace:0:h48399522ee083924
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/backtrace-
backtrace::backtrace::trace_unsynchronized:0:h4499f1decc2119c3
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/backtrace-
1: 0x55e2e05be9c2 - backtrace::backtrace::trace:0:h40e7832466c8123d
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/backtrace-
2: 0x55e2e061df56 - backtrace::capture::Backtrace::create:0:hc9687f6b7ef6261b
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/backtrace-
3: 0x55e2e061de9d - backtrace::capture::Backtrace::new:0:11ae45e25e70ff831
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/backtrace-
4: 0x55e2e0582ec9 - parking_lot_core::parking_lot::deadlock_impl::on_unpark:0:h0b5ec0771432db83
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/parking_lo
5: 0x55e2e0546fab - parking_lot_core::parking_lot::deadlock::on_unpark:0:h75f5f123992865e7
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/parking_lo
6: 0x55e2e05546df - parking_lot_core::parking_lot::park::{{closure}}:0:habf2e4618474516c
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/parking_lo
7: 0x55e2e0551e28 - parking_lot_core::parking_lot::with_thread_data:0:h5eb7ea2dbf01fb62
    at /home/agourlay/.cargo/registry/src/github.com-1ecc6299db9ec823/parking_lo
parking_lot_core::parking_lot::park:0:h34e17314b20ce15d
```

# Static deadlock detector

- [github.com/BurtonQin/lockbud](https://github.com/BurtonQin/lockbud)

lib/collection: possible deadlocks caused by double-readlock ProxySegment #724

 Closed BurtonQin opened this issue on Jun 22, 2022 · 0 comments

 BurtonQin commented on Jun 22, 2022 

### Current Behavior

`deleted_points` is `Arc<RwLock<...>>`

There are two double read locks in `proxy_segment.rs`

The first read lock is on L130 in fn `search`.

fn `add_deleted_points_condition_to_filter` is called on L140.

```
qdrant/lib/collection/src/collection_manager/holders/proxy_segment.rs
Lines 130 to 140 in asof491
130     let deleted_points = self.deleted_points.read();
131
132     // Some point might be deleted after temporary segment creation
133     // We need to prevent them from being found by search request
134     // That is why we need to pass additional filter for deleted points
135     let do_update_filter = !deleted_points.is_empty();
136     let mut wrapped_result = if do_update_filter {
137         // ToDo: Come up with better way to pass deleted points into Filter
138         // e.g. implement AtomicRefCell for Serializer.
139         // This copy might slow process down if there will be a lot of deleted points
140         let wrapped_filter = self.add_deleted_points_condition_to_filter(filter);
```

---

# Going distributed

- Stay available
  - Scaling out on commodity machines
  - Shards & replicas per collection
  - No leader shard for writes
  - Transactional vector operations are opt-in
-



---

# Raft consensus

- Used to synchronize cluster & collection topology
  - Agree on sequence of operations (log)
  - Leader commits after entry replicated by majority
  - raft.rs is not easy but maintainers are responsive
  - Debugging can be very difficult
-

---

# Takeaways

- Demystified vector search engine
  - New indexing schemes (ANN/HNSW)
  - Apply SIMD to bottlenecks if possible
  - Mind the deadlocks
  - Distributed systems are hard
  - Another data point in favor of Rust
-

---

## Farewell links

- <https://github.com/qdrant/qdrant>
- <https://qdrant.tech/documentation>
- <https://qdrant.tech/benchmarks>
- <https://blog.qdrant.tech>
- <https://qdrant.to/discord>

