Векторная база Qdrant: Теоретический анализ и архитектурные принципы

1. Введение: Позиционирование Qdrant в экосистеме векторных баз данных

Qdrant — это высокопроизводительная, распределенная векторная база данных с открытым исходным кодом, написанная на языке Rust. Она спроектирована для production-сценариев, требующих работы с большими объемами данных, и отличается мощными возможностями фильтрации по метаданным и горизонтального масштабирования.

В экосистеме векторных баз данных Qdrant занимает нишу между простыми, встраиваемыми решениями (например, Chroma) и сложными микросервисными платформами (например, Milvus).

- **Переход от Chroma к Qdrant:** Проекты часто начинаются с более простых решений, таких как Chroma, для быстрой разработки MVP (Minimum Viable Product) на одном сервере с низкой нагрузкой (QPS < 30). Однако, по мере роста требований, в частности, при необходимости сложной фильтрации по метаданным (вложенные структуры, комбинации AND/OR, range-фильтры) и горизонтального масштабирования, Qdrant становится логичным следующим шагом.
- **Переход от Qdrant к Milvus:** Для систем со стабильной, но высокой нагрузкой (десятки миллионов векторов, тысячи запросов в минуту), Qdrant является оптимальным выбором благодаря поддержке сложных фильтров и масштабирования коллекций. При переходе к гипермасштабируемым нагрузкам (сотни миллионов векторов, десятки тысяч QPS) и необходимости независимого масштабирования операций чтения (поиска) и записи, может потребоваться переход на микросервисные архитектуры, такие как Milvus.

Данный материал фокусируется на теоретических и математических основах, которые делают Qdrant мощным решением для production-сценариев.

2. Математические основы векторного поиска

Векторные базы данных, включая Qdrant, функционируют в многомерных метрических пространствах. Их основная задача — поиск по сходству (similarity search), формализуемый как задача поиска ближайшего соседа (Nearest Neighbor, NN).

Определение 1: Векторное пространство и эмбеддинги. Пусть $\mathcal{X}$ — пространство исходных объектов (текстов, изображений и т.д.). Эмбеддинг (vector embedding) — это отображение $f : \mathcal{X} \to \mathbb{R}^d$, которое сопоставляет каждому объекту $x \in \mathcal{X}$ вектор $\vec{v}$ в d-мерном вещественном векторном пространстве $\mathbb{R}^d$. Ключевое свойство этого отображения — сохранение семантической близости: если объекты x_1 и x_2 семантически близки, то расстояние между их векторными представлениями $f(x_1)$ и $f(x_2)$ должно быть малым.

Определение 2: Метрики сходства. Сходство между векторами $\vec{u}, \vec{v} \in \mathbb{R}^d$ измеряется с помощью метрики расстояния $d(\vec{u}, \vec{v})$ или функции сходства $S(\vec{u}, \vec{v})$. Qdrant поддерживает несколько ключевых метрик:

1. **Косинусное сходство (Cosine Similarity):**

$$S_C(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_{i=1}^{d} u_i^2} \sqrt{\sum_{i=1}^{d} v_i^2}}$$

   - **Пошаговая деривация:**
     1. $\vec{u} \cdot \vec{v} = \sum_{i=1}^{d} u_i v_i$: Вычисляется скалярное произведение векторов, которое представляет собой сумму произведений их соответствующих компонент.
     2. $\|\vec{u}\| = \sqrt{\sum_{i=1}^{d} u_i^2}$: Вычисляется евклидова норма (длина) вектора $\vec{u}$. Аналогично для $\|\vec{v}\|$.
     3. Результат скалярного произведения делится на произведение длин векторов. Это нормализует результат, делая его независимым от магнитуды векторов.
   - **Практическое применение:** Косинусное сходство измеряет косинус угла между двумя векторами и принимает значения в диапазоне $[-1, 1]$. Оно идеально подходит для задач обработки естественного языка (NLP), где направление вектора (семантическое содержание) важнее его длины. В контексте поиска часто используется косинусное расстояние $d_C(\vec{u}, \vec{v}) = 1 - S_C(\vec{u}, \vec{v})$.

2. **Евклидово расстояние (Euclidean Distance, L2-норма):**

$$d_E(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\|_2 = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2}$$

- **Пошаговая деривация:**
  1. $(u_i - v_i)$: Для каждой размерности i вычисляется разность компонент.
  2. $(u_i - v_i)^2$: Разность возводится в квадрат, чтобы все слагаемые были неотрицательными.
  3. $\sum_{i=1}^{d}(\dots)$: Квадраты разностей суммируются по всем размерностям.
  4. $\sqrt{\dots}$: Извлекается квадратный корень для возврата к исходной единице измерения.
- **Практическое применение:** Эта метрика представляет собой "прямолинейное" расстояние между двумя точками в d-мерном пространстве. Она интуитивно понятна и часто используется в задачах компьютерного зрения.

3. **Скалярное произведение (Dot Product):**

$$S_{DP}(\vec{u}, \vec{v}) = \vec{u} \cdot \vec{v} = \sum_{i=1}^{d} u_i v_i$$

- **Пошаговая деривация:** Каждая компонента u_i вектора $\vec{u}$ умножается на соответствующую компоненту v_i вектора $\vec{v}$, и все произведения суммируются.
- **Практическое применение:** Скалярное произведение напрямую связано с косинусным сходством ($S_{DP} = \|\vec{u}\|\|\vec{v}\|S_C$). Если все векторы в базе данных предварительно нормализованы до единичной длины ($\|\vec{v}\| = 1$), то скалярное произведение становится эквивалентным косинусному сходству. Qdrant может использовать этот факт для значительного ускорения вычислений, так как умножение и сложение выполняются быстрее, чем операции деления и извлечения корня, необходимые для косинусного сходства.

Определение 3: Задача поиска приблизительного ближайшего соседа (Approximate Nearest Neighbor, ANN). Точный поиск NN в пространствах высокой размерности вычислительно неэффективен из-за "проклятия размерности". Поэтому на практике решается задача ANN. Пусть $P \subset \mathbb{R}^d$ — набор векторов, а $q \in \mathbb{R}^d$ — вектор запроса. Задача состоит в том, чтобы найти вектор $p' \in P$, удовлетворяющий условию:

$$d(q, p') \leq (1 + \varepsilon) \min_{p \in P} d(q, p)$$

где $\varepsilon > 0$ — параметр, определяющий допустимую погрешность, а $\min_{p \in P} d(q, p)$ — расстояние до истинного ближайшего соседа. Qdrant использует индексирующие структуры для эффективного решения этой задачи.

---

3. Основная структура индексации: Hierarchical Navigable Small World (HNSW)

HNSW — это основной алгоритм, используемый в Qdrant для решения задачи ANN. Он основан на создании многоуровневого графа близости и известен своей высокой производительностью и возможностью динамического добавления элементов без полной перестройки индекса.

Структура HNSW: Алгоритм строит иерархию графов $G_0, G_1, \dots, G_L$. `layer 2 (entry layer) •————————• (entry point)`
`| layer 1 •——•——•——• | | | layer 0 •–•–•–•–•–•–•–•`

- **Базовый слой (G_0):** Содержит все векторы. Каждый узел соединен с несколькими своими ближайшими соседями, образуя граф со свойством "маленького мира" (small world).
- **Верхние слои ($G_l, l > 0$):** Каждый слой G_l является разреженным подграфом слоя G_{l-1}. Узлы для верхних слоев выбираются вероятностно, что обеспечивает логарифмическое уменьшение числа узлов с ростом номера слоя. Верхние слои содержат "длинные" ребра, позволяющие быстро перемещаться по пространству, в то время как нижние слои с "короткими" ребрами обеспечивают точность локального поиска.

Алгоритм вставки нового элемента q:

1. **Выбор максимального слоя:** Для q случайным образом выбирается максимальный слой l_{max}, на котором он будет присутствовать, с экспоненциально убывающей вероятностью.
2. **Поиск точки входа:** Поиск начинается с единственной точки входа на самом верхнем слое L.
3. **Итеративный спуск:** На каждом слое l от L до $l_{max} + 1$ выполняется жадный поиск ближайшего к q элемента. Найденный на слое l узел становится точкой входа для поиска на слое $l - 1$.
4. **Вставка и соединение:** На каждом слое l от $\min(L, l_{max})$ до 0 выполняется поиск M ближайших соседей для q. Между q и найденными соседями устанавливаются ребра. Применяются эвристики для ограничения максимального числа связей у каждого узла.

Алгоритм поиска k ближайших соседей к запросу q:

1. **Поиск точки входа:** Аналогично вставке, поиск начинается с верхнего слоя.
2. **Жадный спуск:** На слоях с L до 1 выполняется жадный поиск для нахождения узла, ближайшего к q, который

служит точкой входа для следующего, более плотного слоя.
3. **Поиск на базовом слое:** На слое G_0 выполняется более тщательный поиск (например, beam search) для нахождения k ближайших соседей.

Анализ сложности и ограничения: Средняя сложность поиска в HNSW составляет $O(\log N)$, где N — число векторов. Однако важно понимать ограничения *классического* алгоритма HNSW:

- **Фильтрация:** Алгоритм в его базовой форме не поддерживает эффективную префильтрацию. Попытка выполнить поиск по отфильтрованному подмножеству данных (filtered search) часто приводит к деградации производительности.
- **Квантование и Шардинг:** HNSW как алгоритм не включает в себя механизмы продуктового квантования (Product Quantization) или шардинга.

Qdrant решает эти ограничения, реализуя данные механизмы на уровне всей системы, поверх своей кастомизированной реализации HNSW.

---

4. Оптимизации и уникальные механизмы Qdrant

Qdrant расширяет стандартный HNSW, внедряя собственные механизмы для оптимизации производительности, памяти и функциональности.

4.1. Квантование (Quantization)

Квантование сокращает точность представления векторов для уменьшения занимаемой памяти и ускорения вычислений. Qdrant поддерживает скалярное и продуктовое квантование.

A. Скалярное квантование (Scalar Quantization, SQ): Преобразует 32-битные числа с плавающей запятой (float32) в 8-битные целые числа (int8).

- **Математическая деривация:** Пусть дан вектор $\vec{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$.
    1. **Определение диапазона:** Для набора данных находятся значения, определяющие диапазон, например, 2-й и 98-й перцентили (p_2 и p_{98}), для устойчивости к выбросам.
    2. **Вычисление шага квантования:** Для b-битного квантования (обычно $b=8$, что дает $2^8 = 256$ уровней) шаг Δ вычисляется как:
    $$\Delta = \frac{p_{98} - p_2}{2^b - 1}$$
    3. **Квантование:** Каждая компонента v_i преобразуется в целое число v'_i:
    $$v'_i = \text{round}\left(\frac{v_i - p_2}{\Delta}\right)$$
    где $\text{round}(\cdot)$ — функция округления.
    4. **Де-квантование:** Приближенное исходное значение $\hat{v}_i$ можно восстановить:
    $$\hat{v}_i = v'_i \cdot \Delta + p_2$$
- **Практическое применение:** Qdrant может вычислять метрики расстояния непосредственно на квантованных векторах, используя быстрые целочисленные SIMD-инструкции (например, AVX). Это снижает потребление RAM в 4 раза и ускоряет вычисления расстояний, что критически важно для производительности. Вносимая ошибка квантования приемлема для большинства задач, включая RAG (Retrieval-Augmented Generation).

B. Продуктовое квантование (Product Quantization, PQ): Обеспечивает еще более сильное сжатие, особенно для хранения векторов на диске.

- **Математическая деривация:**
    1. **Разбиение вектора:** Вектор $\vec{v} \in \mathbb{R}^d$ разбивается на m суб-векторов $\vec{v}_1, \ldots, \vec{v}_m$ размерности d/m.
    2. **Создание кодовых книг:** Для каждого из m подпространств с помощью k-means создается своя кодовая книга (codebook) C_j, состоящая из k центроид (обычно $k=256$).
    3. **Кодирование:** Каждый суб-вектор $\vec{v}_j$ заменяется индексом i_j ближайшего к нему центроида из книги C_j. Вектор $\vec{v}$ представляется набором из m индексов $(i_1, \ldots, i_m)$.
- **Практическое применение (Asymmetric Distance Computation):** Для вычисления расстояния между вектором

запроса q и сжатым вектором v, q разбивается на суб-векторы $(q_1, \ldots, q_m)$. Для каждого подпространства j предварительно вычисляется таблица расстояний между q_j и всеми k центроидами из C_j. Итоговое расстояние аппроксимируется как сумма предвычисленных значений из этих таблиц. Этот метод чрезвычайно быстр. Qdrant использует PQ для хранения векторов на диске, что позволяет работать с наборами данных, значительно превышающими объем RAM.

4.2. Механизм фильтрации

Способность Qdrant эффективно сочетать ANN-поиск с фильтрацией по метаданным (payload) является одним из его ключевых преимуществ. Это решает задачу **фильтрованного ANN-поиска** и является основной причиной для выбора Qdrant вместо более простых баз данных.

Определение 5: Инвертированный индекс (Inverted Index). Для быстрой фильтрации Qdrant строит инвертированный индекс по полям метаданных. Этот индекс отображает значения полей в список ID векторов, обладающих этими значениями. Например: `{"color": "blue"}` -> `[id_1, id_5, id_42, ...]`. Qdrant поддерживает сложные фильтры, включая вложенные JSON-структуры, числовые диапазоны, геолокацию и сложные логические комбинации.

Решение Qdrant: Pre-filtering Наивный подход "post-filtering" (сначала найти k соседей, затем отфильтровать) неэффективен и часто возвращает пустой результат. Qdrant реализует "pre-filtering":

1. С помощью инвертированного индекса формируется множество ID векторов S_{filter}, удовлетворяющих условию фильтра.
2. ANN-поиск по графу HNSW выполняется таким образом, что рассматриваются только узлы, принадлежащие множеству S_{filter}.

Ключевая инновация Qdrant заключается в том, что его реализация HNSW эффективно работает с такими разреженными наборами ID. Во время обхода графа, если узел не принадлежит S_{filter}, он игнорируется, но его соседи все равно могут быть рассмотрены. Это позволяет поддерживать связность поиска и избегать катастрофического падения производительности, характерного для многих других реализаций HNSW при фильтрации.

---

5. Надежность и персистентность

Write-Ahead Log (WAL): Qdrant обеспечивает долговечность данных (Durability из ACID) с помощью журнала упреждающей записи.

- **Принцип:** Любая операция модификации (вставка, удаление) сначала записывается в лог на диске и только потом применяется к структурам в памяти.
- **Гарантия:** В случае сбоя состояние системы восстанавливается путем воспроизведения записей из WAL поверх последнего консистентного снимка (snapshot), что гарантирует отсутствие потерь подтвержденных записей.

Динамические обновления и удаления: Удаление узла из HNSW-графа — сложная операция.

- **Решение Qdrant:** Вместо физического удаления узел помечается как удаленный (например, с помощью битовой маски) и игнорируется при поиске. Физическое удаление и реорганизация графа происходят асинхронно в фоновом процессе оптимизации (compaction). Это обеспечивает высокую пропускную способность операций записи и удаления без блокировки чтения.

---

6. Заключение: Синтез теоретических принципов

Qdrant представляет собой сложную систему, построенную на синергии фундаментальных концепций:

- **Теория графов и алгоритмы:** Кастомизированная реализация HNSW для эффективного ANN-поиска с логарифмической сложностью.
- **Теория информации:** Скалярное и продуктовое квантование для сжатия данных и ускорения вычислений, что позволяет работать с большими объемами данных как в RAM, так и на диске.
- **Теория баз данных:** Инвертированные индексы для мощной и быстрой фильтрации по метаданным, а также WAL для обеспечения надежности и персистентности.

Уникальность Qdrant заключается в глубокой интеграции этих алгоритмов. Способность эффективно выполнять предварительную фильтрацию в HNSW, сочетая ее с многоуровневым квантованием и надежной моделью хранения, выделяет его как высокопроизводительное и функционально богатое решение для задач векторного поиска в промышленных масштабах. Это делает его оптимальным выбором для систем, переросших базовые решения и требующих сложной логики и масштабируемости, но еще не достигших уровня гипермасштабирования, требующего

микросервисных архитектур.