

Векторный поиск в ClickHouse





Филатенков Артур



**Saint
HighLoad⁺⁺**



Команда разработчиков

- Филатенков Артур ( [FArthur-cmd](#))
- Евсюков Никита ( [NikitaEvs](#))
- Макаров Владимир ( [VVMak](#))
- Печенкин Александр ( [Piachonkin-Alex](#))
- Мишин Данила ( [JungleTryne](#))
- Василенко Никита ( [NikeVas](#))
- Сагателян Акоп ( [HACKONDEX](#))

Что такое векторный поиск?



Время выполнения запроса

```
SELECT id FROM table ORDER BY L2Distance(column, x) LIMIT N
```

Размерность пространства – 256

10000000 rows in set. Elapsed: 23.699 sec. Processed 10.00 million rows, 10.40 GB (421.96 thousand rows/s., 438.84 MB/s.)

Размерность пространства – 512

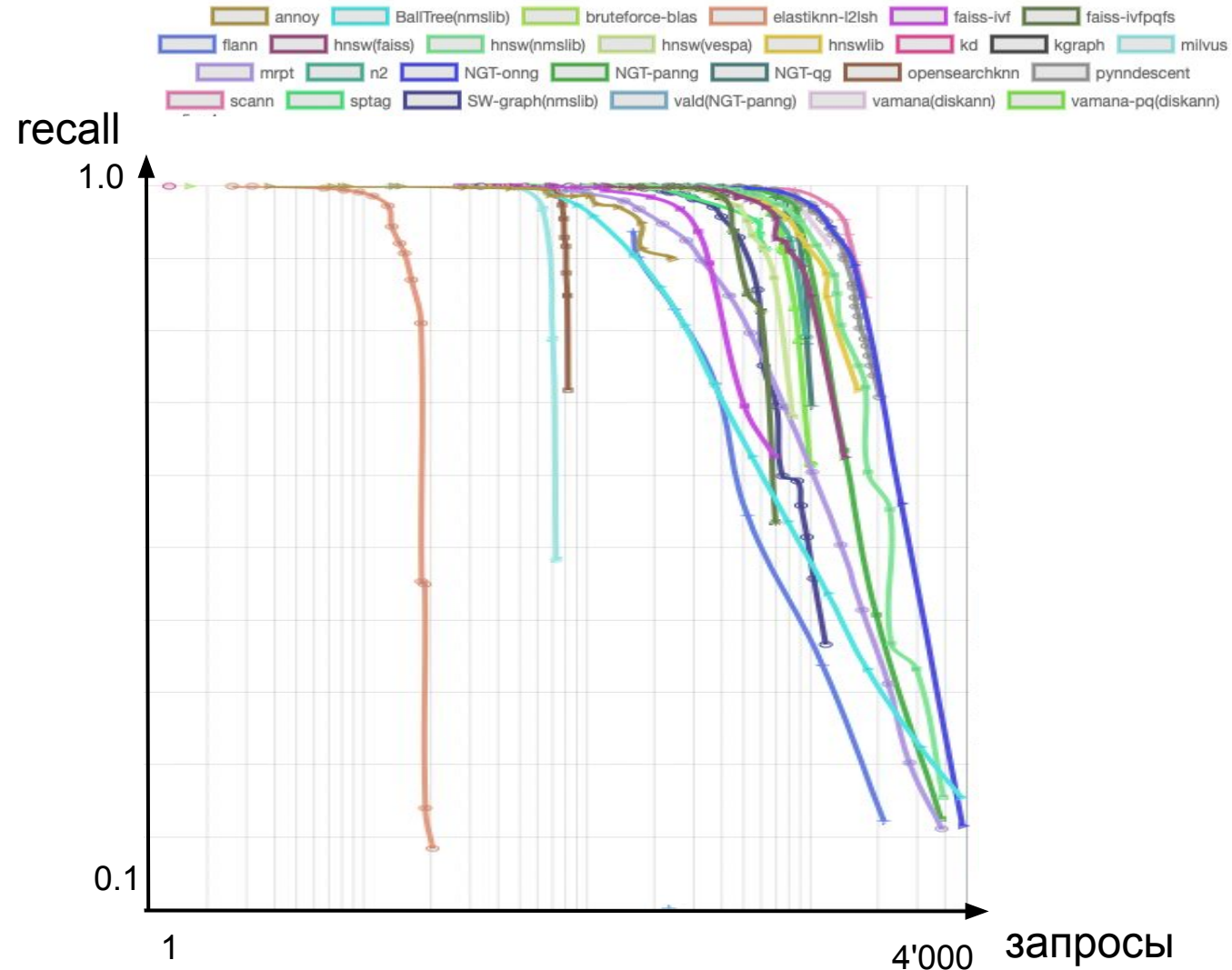
10000000 rows in set. Elapsed: 40.948 sec. Processed 10.00 million rows, 20.64 GB (244.21 thousand rows/s., 504.05 MB/s.)

Что можно сделать?

- Быстрее работающие функции расстояния
- Хранить меньше промежуточных данных

Существующие реализации

- HNSW
- Faiss
- DiskANN
- Annoy
- ScaNN



Коротко о выбранных

→ HNSW – известный алгоритм

Коротко о выбранных

- ✓ HNSW – известный алгоритм
- Annoy – библиотека Spotify

Коротко о выбранных

- ✓ HNSW – известный алгоритм
- ✓ Annoy – библиотека Spotify
- ➔ Faiss – библиотека Facebook (комбинации алгоритмов)

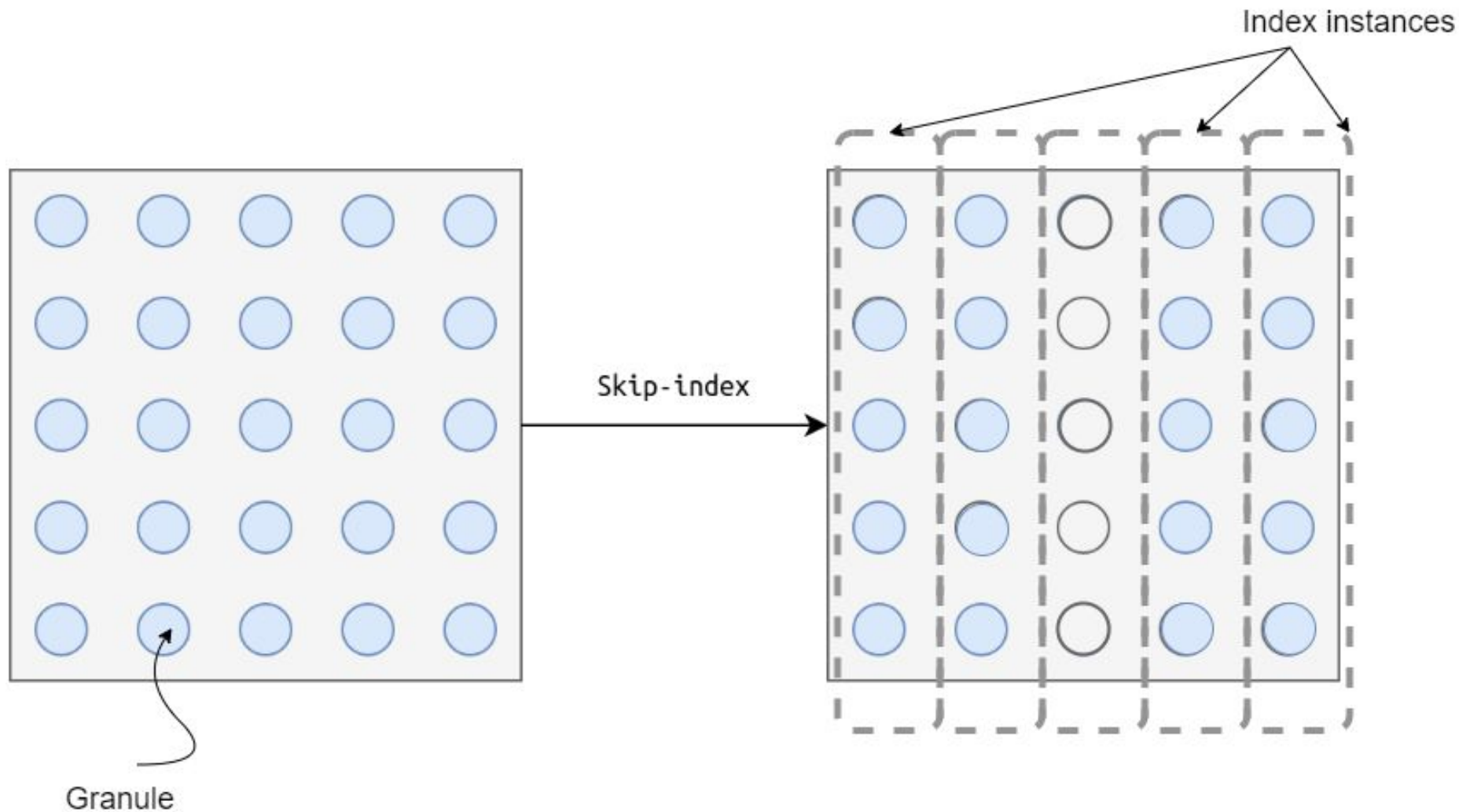
Коротко о выбранных

- ✓ HNSW – известный алгоритм
- ✓ Annoy – библиотека Spotify
- ✓ Faiss – библиотека Facebook (комбинации алгоритмов)
- ➔ DiskANN – библиотека Microsoft (NSG)

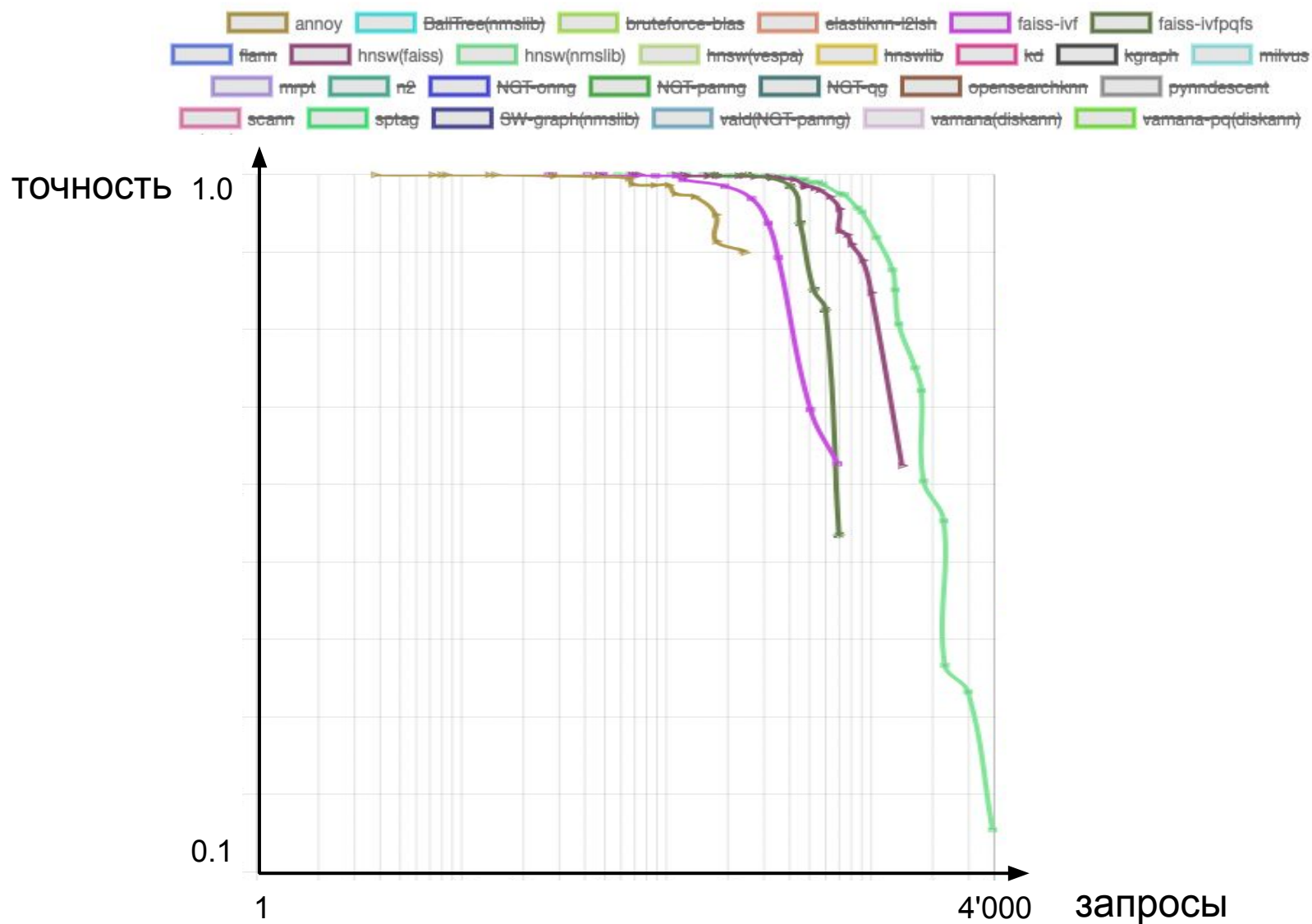
Коротко о выбранных

- ✓ HNSW – известный алгоритм
- ✓ Annoy – библиотека Spotify
- ✓ Faiss – библиотека Facebook (комбинации алгоритмов)
- ✓ DiskANN – библиотека Microsoft (NSG)
- ➔ ScaNN – алгоритм Google

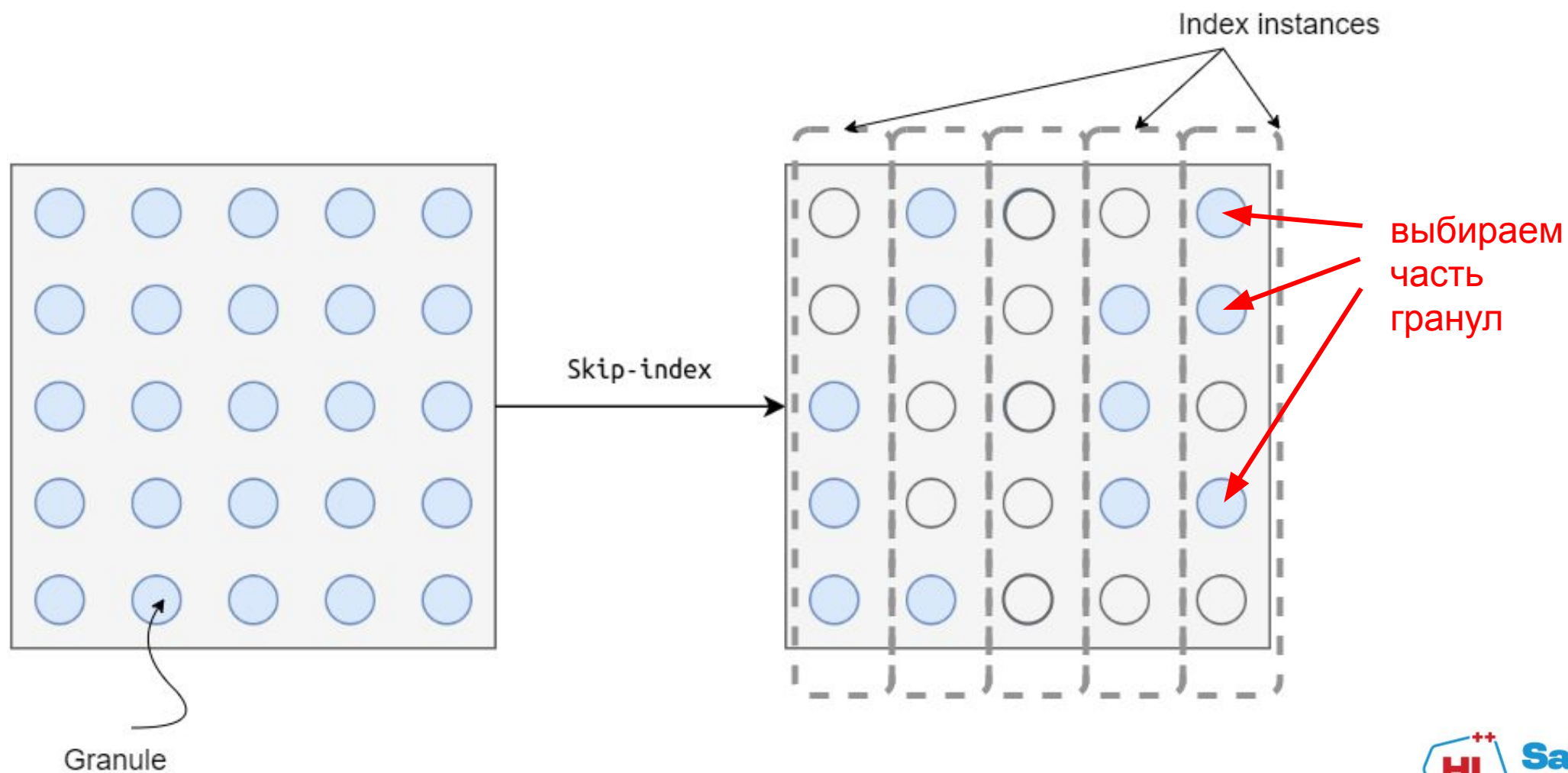
Как добавить новые индексы?



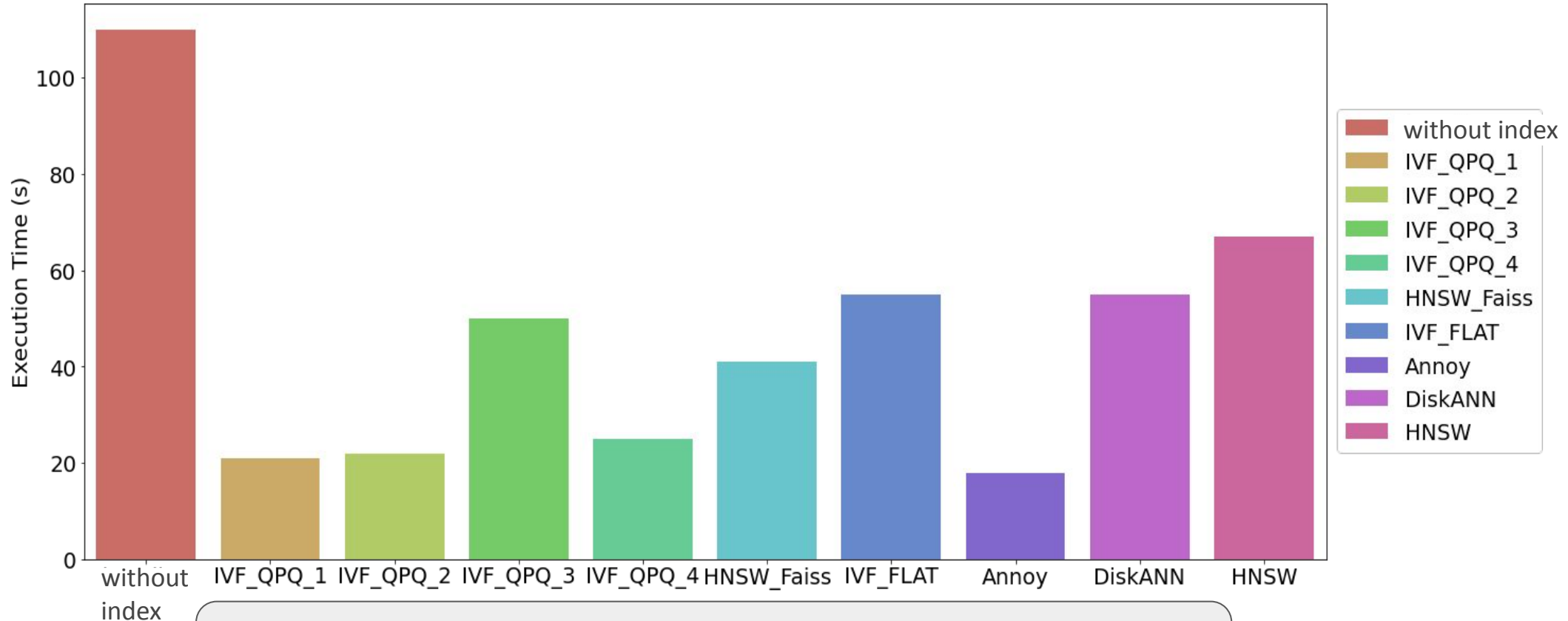
Приближенность индексов



Approximate Nearest Neighbour Index



Первые успехи



```
SELECT url FROM table ORDER BY L2Distance(embedding, x) LIMIT N
```

В чем же подвох?

- Scann не был добавлен
- Некоторые библиотеки все еще не добавлены
- Нетривиальная зависимость от гиперпараметров
- Изначальная реализация для Tuple
- recall 6%

Проблемы сборки

- Зависимости библиотек усложняют интеграцию



Tuple или Array?

- Хранение Tuple неоптимально для данной задачи

Array:

100 rows in set. Elapsed: 75.262 sec. Processed 9.65 million rows, 20.88 GB (128.18 thousand rows/s., 277.39 MB/s.)

Tuple:

100 rows in set. Elapsed: 291.636 sec. Processed 9.65 million rows, 20.88 GB (33.08 thousand rows/s., 71.32 MB/s.)

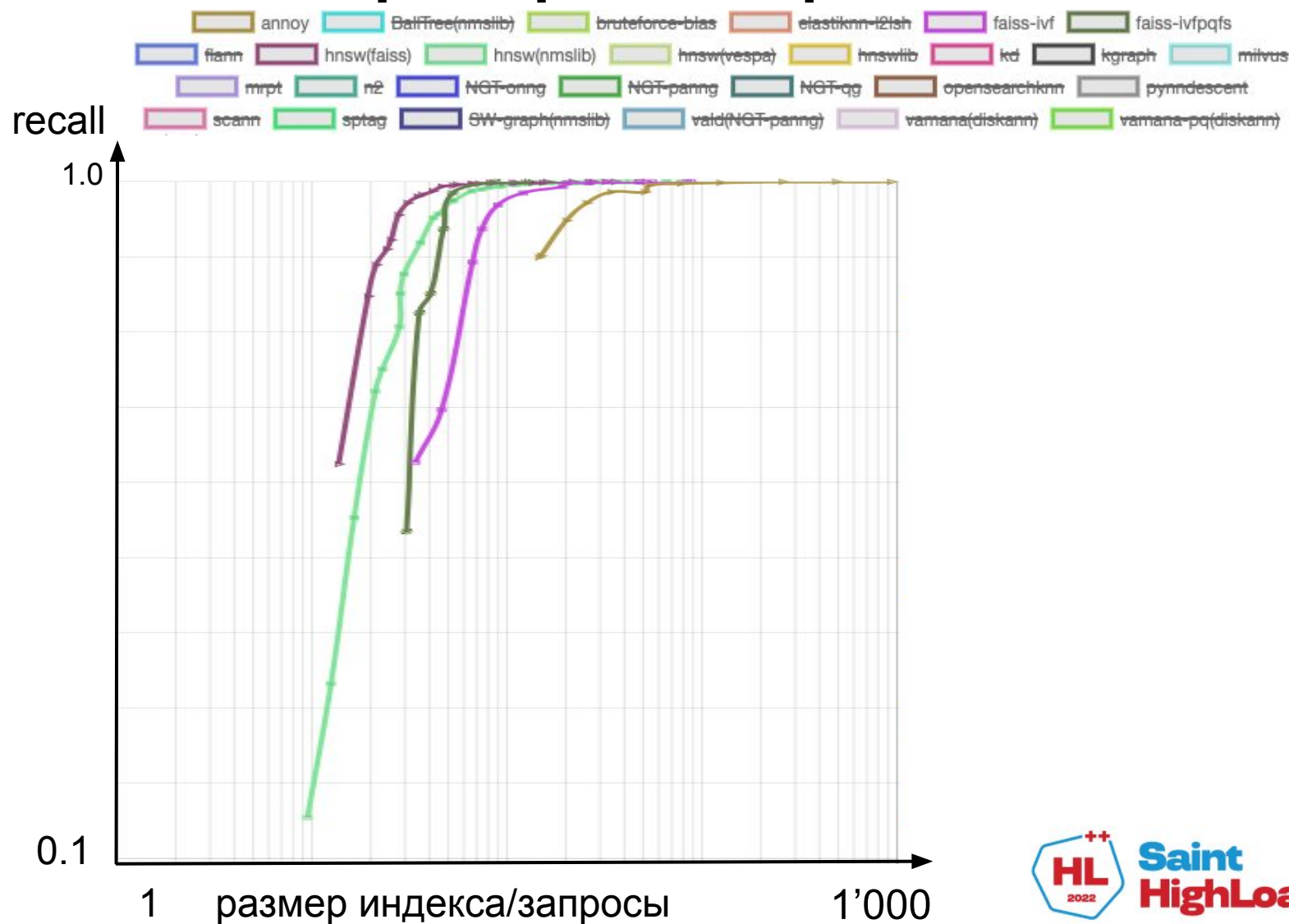
Зависимость от гиперпараметров

- Размер индекса влияет на скорость запросов

data – 19G

hnsw – 38G

annoy – 3.2G



Слишком приближенные?

- Recall 6%

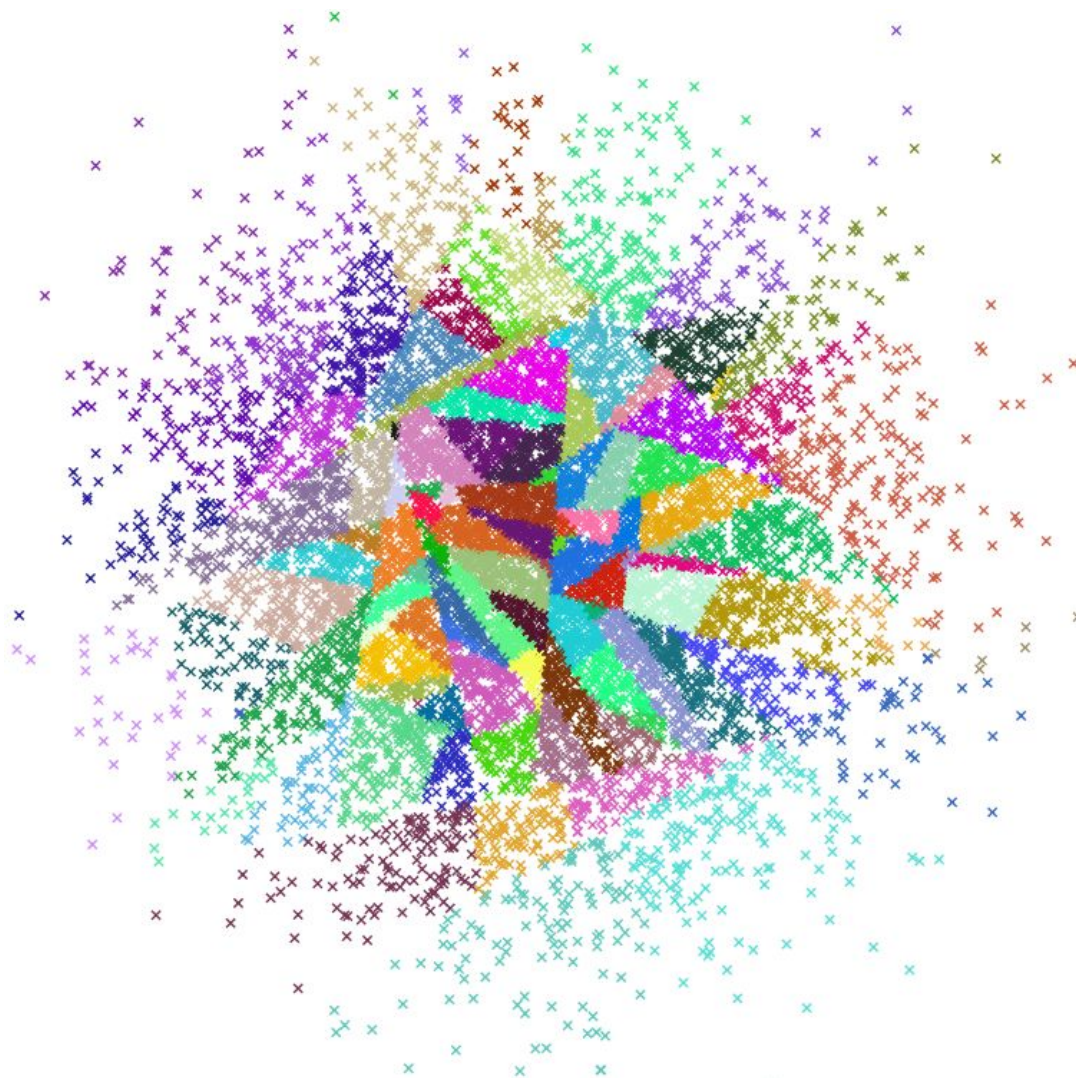
Слишком приближенные?

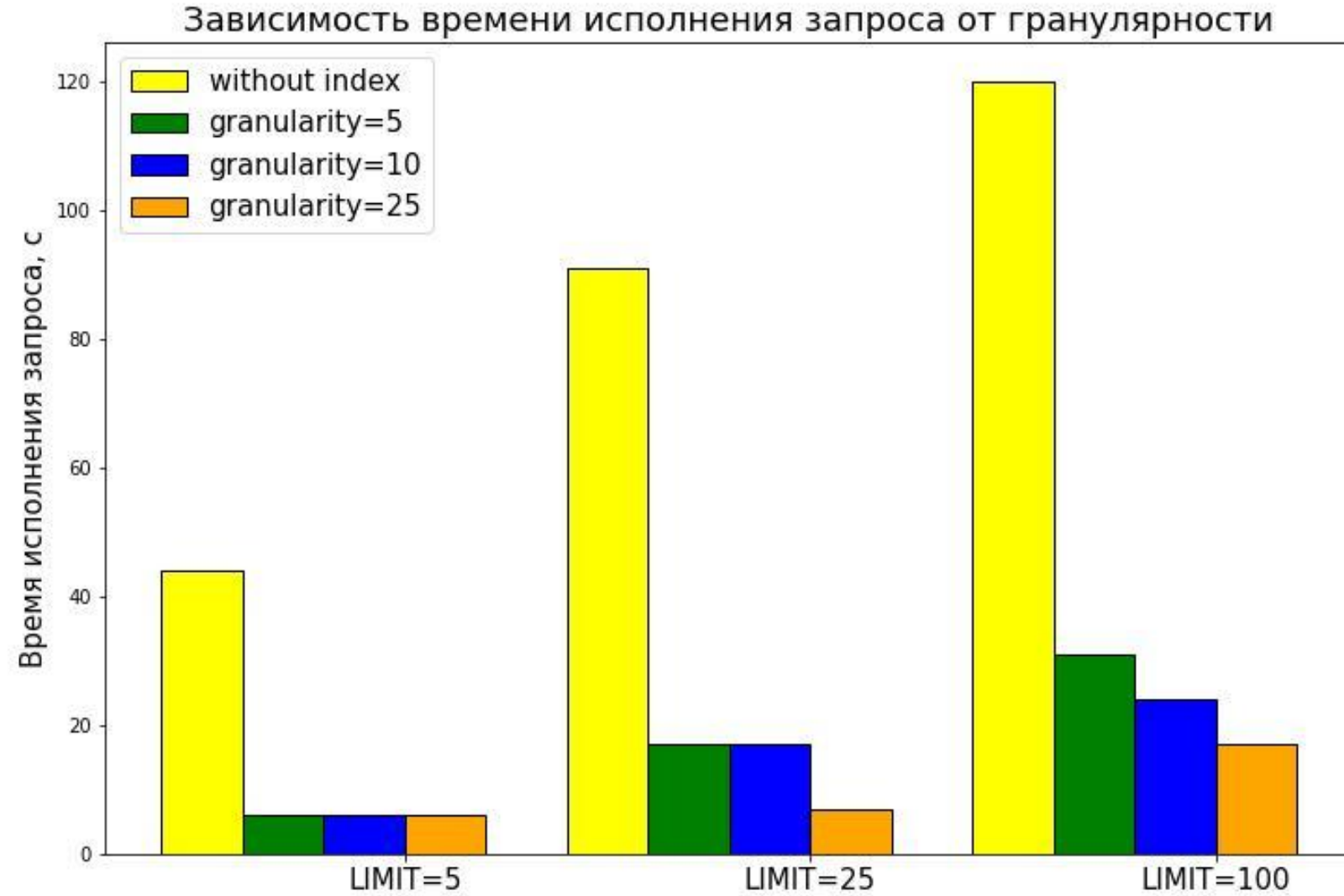
- Recall 6%



Что доступно уже сейчас?

Annoy





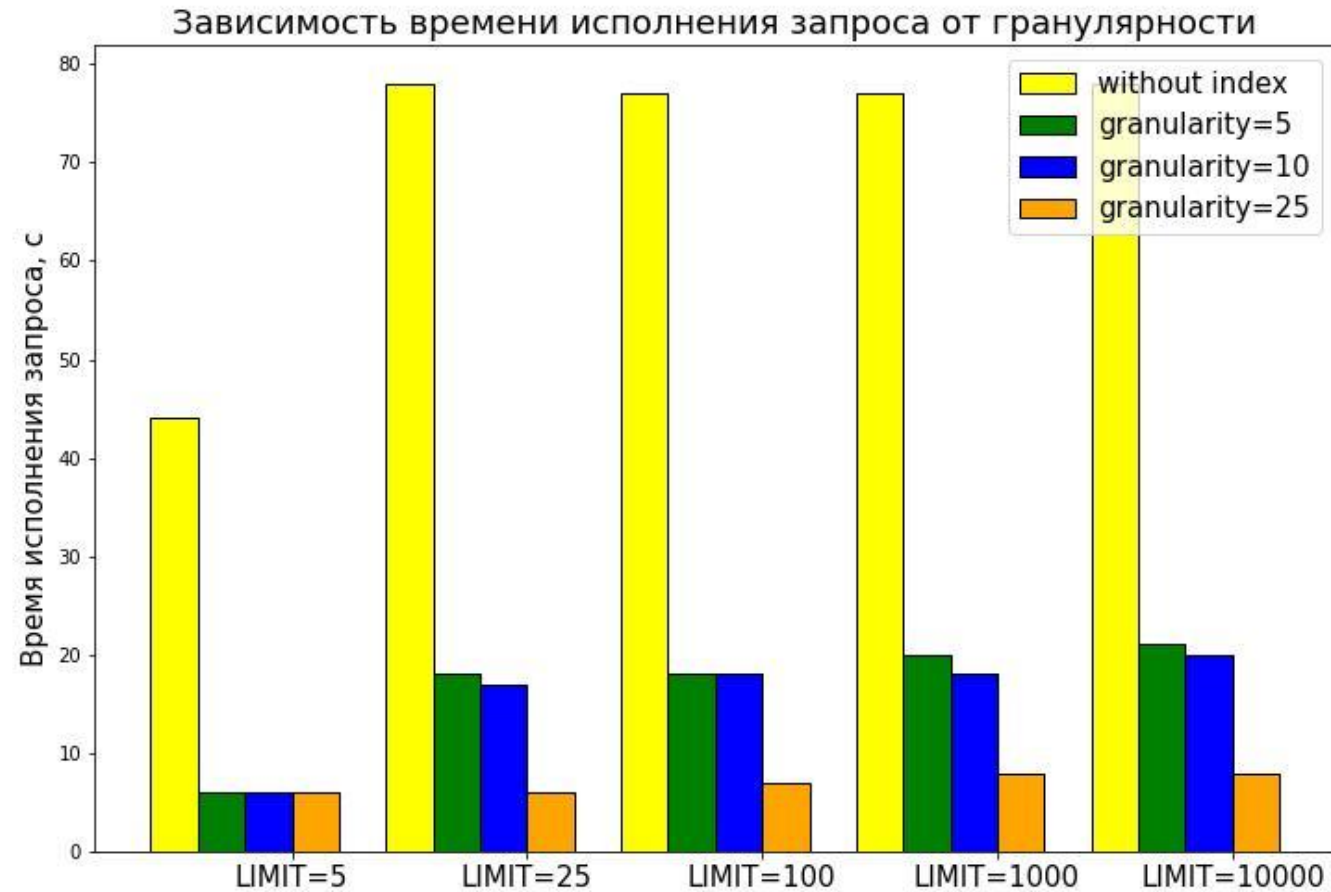
Потребляемая память

Исходный размер данных: 50G

Исходный размер данных после сжатия: 20G

Значение параметра	Объем индекса, G
100	1.2
200	2.0
250	3.6
1000	12.0

Точность и скорость



Recall запросов не превышает 7%

Котики распознаны успешно!



А что если будет не кот?



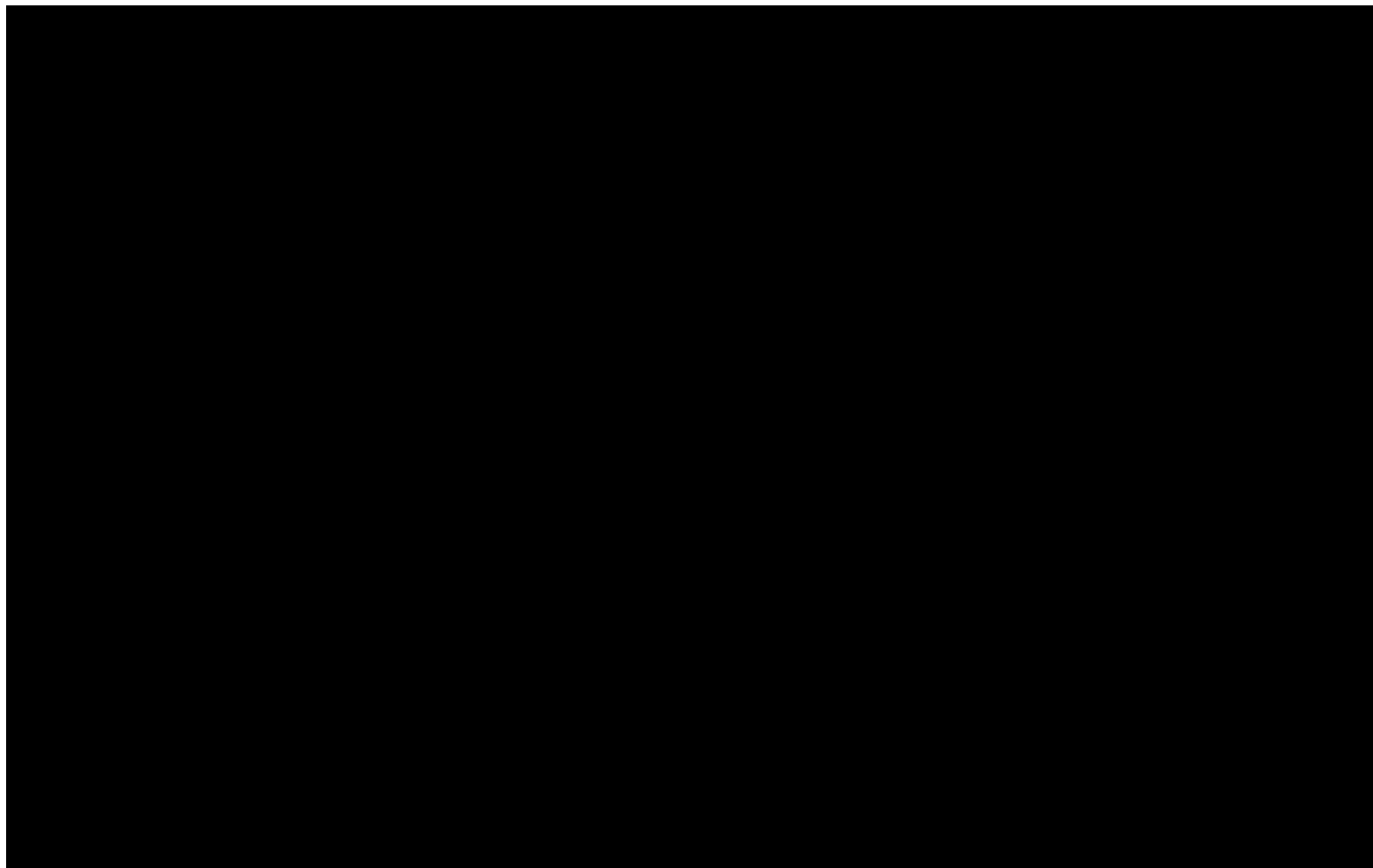
Небольшие замечания

- Важно качество embeddings
- Вставки дольше

Можно использовать

- Распознавание лиц
- Поиск плагиата
- Рекомендательные системы
- Поиск похожих аудио, текстов и картинок

Демонстрация индексов



Посмотреть код



Оценить доклад



**Saint
HighLoad⁺⁺**

