



R&D-центр Т-Банка

**Зачем финтеху распределённые
вычисления и наука**

Алексей Теплов
Архитектор @alekseyteplov

Алексей Теплов

Т-Банк R&D Центр

Архитектор в RnD Центре Т-Банк

Руководжу исследованиями в области систем обработки и хранения данных и возглавляю лабораторию Т в НИУ ВШЭ

Ранее:

Principal Engineer Huawei R&D

Архитектор Центра Развития Банка России

Научный сотрудник НИВЦ МГУ им. М.В. Ломоносова

к.ф.-м.н.

Занимался исследованием масштабируемости распределенных вычислений и ко-дизайном в НИВЦ МГУ



Инженерный R&D

Обсудим

- 01** Что такое R&D в интерпретации Т-Банка
- 02** Зачем вообще индустрии финтеха нужны исследования
- 03** Примеры исследований которые уже работают
- 04** Вызовы которые стоят перед нами в вычислениях

**Что такое R&D в
интерпретации Т -Банка**



R&D Т-Банка



Инженерный R&D

Говорим про компьютерные технологии:

computer science, AI, ML, software engineering

Улучшение технологий

Разработка новых технологий



Продвинутые алгоритмы

Новые методы и ML-модели



Theoretical computer science



Примеры исследований



Что там в ЭТИХ ЦОДах ?

01

**Storage платформа T-Object
и Data Platform**

02

Runtime k8сплатформа Spirit

03

ML Computeплатформа ML Core

Проблемы платформ

01 Data platform

Масштаб

Рост данных и пользователей

Оркестрация между ЦОДами

Жесткий SLA

02 Spirit

Масштаб

Утилизация ресурсов

Управляемость

МежЦОД и Freecooling

03 ML Core

Масштаб

Утилизация гетерогенных
GPU

Коммуникационные сети

Охлаждение и электричество

Утилизация вычислительной инфраструктуры

Утилизация вычислительных ресурсов

Spirit Compute

CPU-серверы в Kubernetes для нагрузок компании.

Непредсказуемые нагрузки требуют держать до 30% мощностей в резерве.

Приложениям нужно многократное резервирование для надёжности. Это увеличивает расход ресурсов.

ML Core

Специализированные GPU-серверы для ML.

Задачи: повысить утилизацию, оптимизировать инференс и обучение моделей.

Гетерогенные ускорители в разных ЦОДах и фрагментация ресурсов усложняют совместную работу.

Общие вызовы: надёжность, прогнозируемость и утилизация

Надёжность

Server Faults Prediction

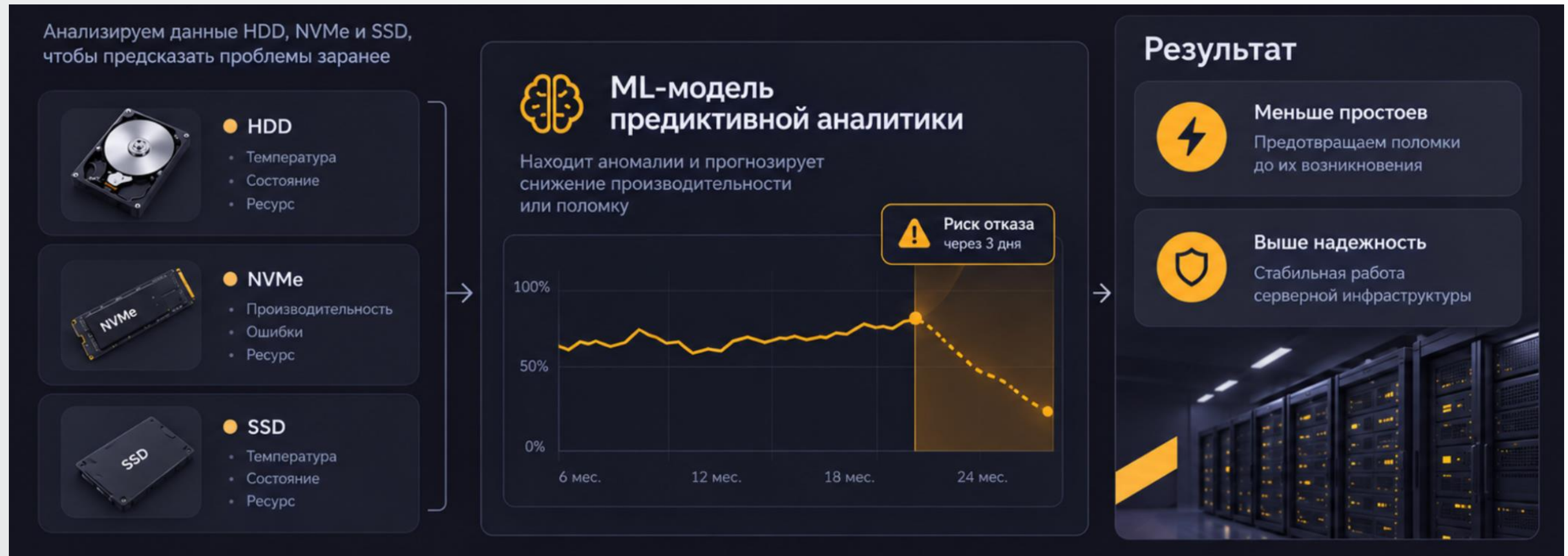
T-Банк R&D x НИВЦ МГУ



×



Цель проекта – разработать и апробировать методы предсказания выхода аппаратного обеспечения серверного оборудования из строя



Server Faults Prediction

Seagate						
	Precision	Recall	PR AUC	ROC AUC	F1	TTF
2020	0,68	0,56	0,54	0,80	0,62	57
2021	0,74	0,69	0,72	0,86	0,72	57
2022	0,80	0,72	0,77	0,90	0,76	61
2023	0,79	0,73	0,76	0,89	0,76	54
2024	0,90	0,72	0,81	0,92	0,80	41
2025	0,86	0,80	0,85	0,93	0,83	58
Toshiba						
	Precision	Recall	PR AUC	ROC AUC	F1	TTF
2023	0,74	0,34	0,45	0,69	0,47	38
2024	0,70	0,71	0,65	0,85	0,70	59
2025	0,70	0,61	0,65	0,88	0,65	51
HGST						
	Precision	Recall	PR AUC	ROC AUC	F1	TTF
2023	0,69	0,77	0,63	0,85	0,73	65
2024	0,63	0,81	0,68	0,93	0,71	90
2025	0,64	0,75	0,64	0,94	0,69	66

- Точность предсказания в целом высокая
- Основное исключение – Toshiba в 2023 году (было введено много дисков новой модели MG08ACA16TA)
- Нет корреляции между размером датасетов и качеством предсказания
- Есть корреляция между годом и качеством (связано в том числе с политикой работы Backblaze)

Проведение инференса

8 моделей дисков Т-Банка
Более 100 дисков каждой модели

 HDD
ML-решения по вендорам: одно для двух моделей Seagate и одно для модели Toshiba

 NVMe
Пять ML-решений, по одному для каждой популярной модели

Тип	Название модели	Число дисков
HDD	ST16000NM000J-2TW103	6192
HDD	ST8000NM017B-2TJ103	2515
NVMe	Intel/Solidigm SSDPF2KE032T1	1006
NVMe	INSPUR-NS8500G2U384	688
HDD	TOSHIBA MG08ADA800E	584
NVMe	INSPUR-NS8600G2U320	560
NVMe	DAPUSTOR DPRD31016TT515T3000	138
NVMe	SAMSUNG MZQL23T8HCLS-00A07	120

Выдача результатов

Результаты каждый день сохраняются наЗ

serial_num	xgb_score	isfo_score	rdate_score	rdisk_score	rmclass_score	stack_score
1 ZA162RXB	0.68126060	0.1261777494076499200	0.8358922484903053	0.121571682447453260	1.7536915137175964	0.260883739542114500
2 ZA162EMX	0.69639385	0.0590398027659211340	0.8358922484903053	0.063815254594372540	1.8407489309045109	0.232179146935699580
3 ZA161SN7	0.65501030	0.0357288983803648040	0.8358922484903053	0.142238992077755870	1.8523671961293902	0.198703179793250420
4 ZA162Y9H	0.50241715	-0.0048183293740270420	0.8158374607638883	0.034360661949986215	1.7967351344973763	0.111564908616505180
5 ZA162QGQ	0.47110656	-0.0036740725188924017	0.8158374607638883	0.111099769344377760	1.7672370631484000	0.101345898796464200
6 WRS118AZ	0.71484470	-0.0759374370037938000	0.5393403991624508	0.215361448621439400	1.9386051640107590	0.091212733359518810
7 WRS118B6	0.59151910	-0.0650948320151655800	0.5393403991624508	0.526447142408640700	2.1835797743745666	0.084925896830149340
8 WRS09PCP	0.57036080	-0.0646617629525106400	0.5393403991624508	0.527434928123892700	2.1869965490592826	0.080868079250825290
9 WRS0DJCZ	0.58117960	-0.0758397680338777100	0.5004909801116089	0.403659712489124860	2.1529810024573837	0.071297226623282280
10 WRS09GPH	0.56601244	-0.0741036261772612400	0.5004909801116089	0.403659712489124860	2.1814856815806830	0.070675509873848630
11 WRS0EA8Y	0.56420210	-0.0757041159214633100	0.5004909801116089	0.403659712489124860	2.1915535630892800	0.070505657784773120
12 WRS09HLN	0.53299940	-0.0734867865241432000	0.5004909801116089	0.419132276051161930	2.2641396299792413	0.069685875232294770

Пример выдачи с ранжированием дискова выбранный день

Распределённые приложения

Переезд сервисов в Т-цод вместе с T-Ready

Сопровождение бизнес-линий

Консультации, ревью архитектуры и валидация решений

Связь БЛ с платформами и инфраструктурой

Снятие системных блокеров

Готовность к отказам

Практики отказоустойчивости и эксплуатации в геораспределенной среде

Масштабирование в Т-ЦОД

Помогаем с переносом данных и трафика в регионы

Лаборатории и эксперименты

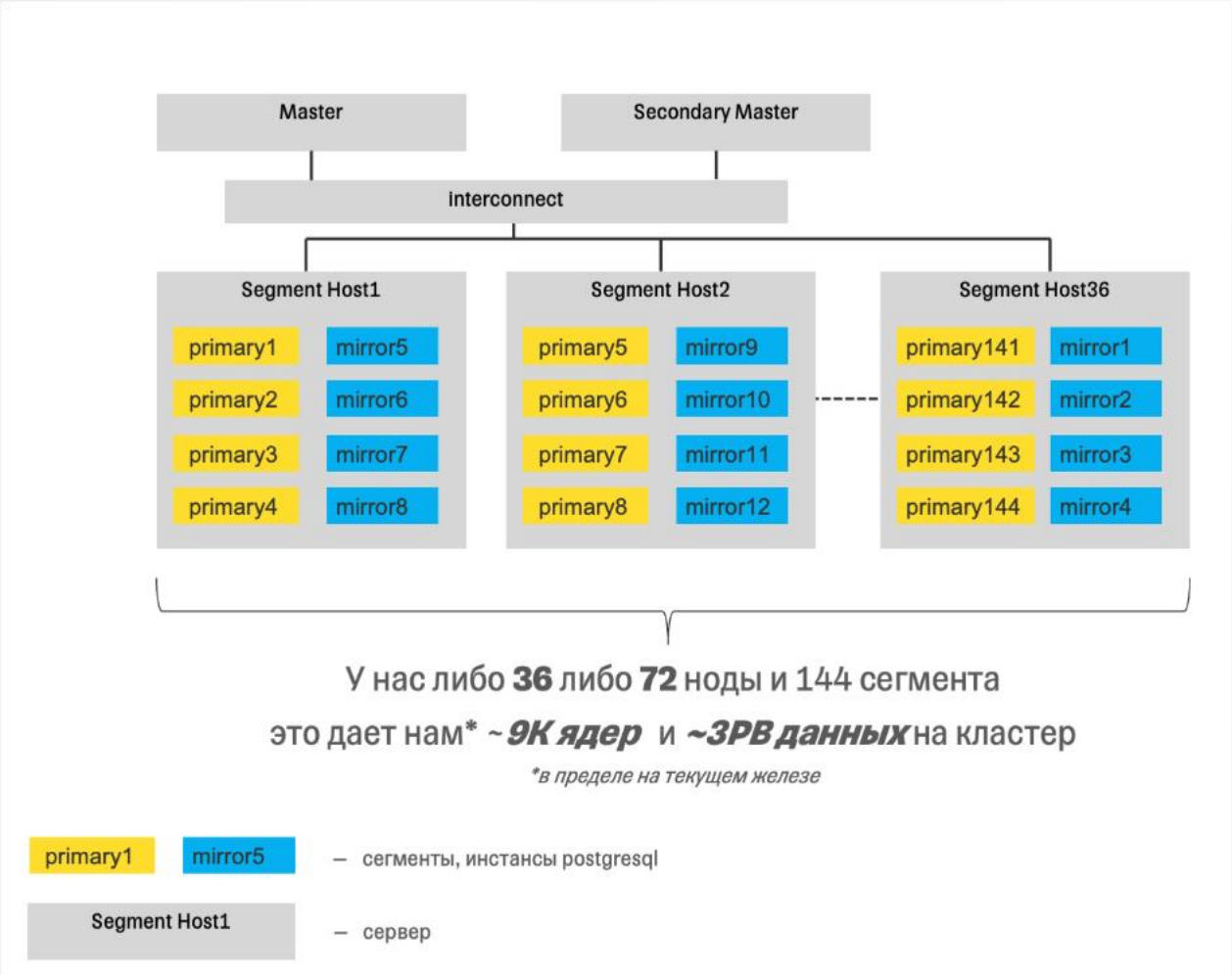
Проверка гипотез и региональных сценариев

Путь подготовки систем

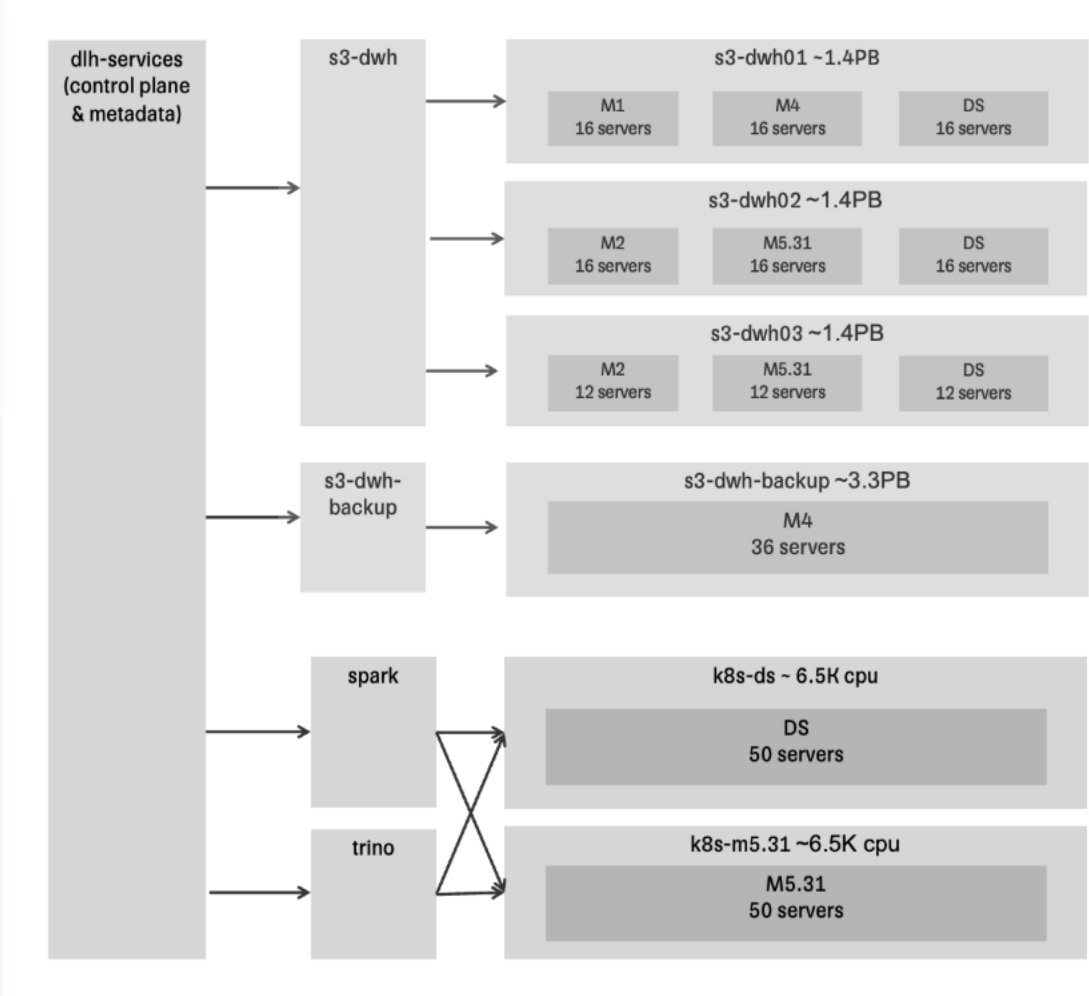
Методология, требования и план технической готовности

Из GreenPlum (GP) в DataLakeHouse (DLH)

Решение проблемы масштабируемости через переход на новую архитектуру



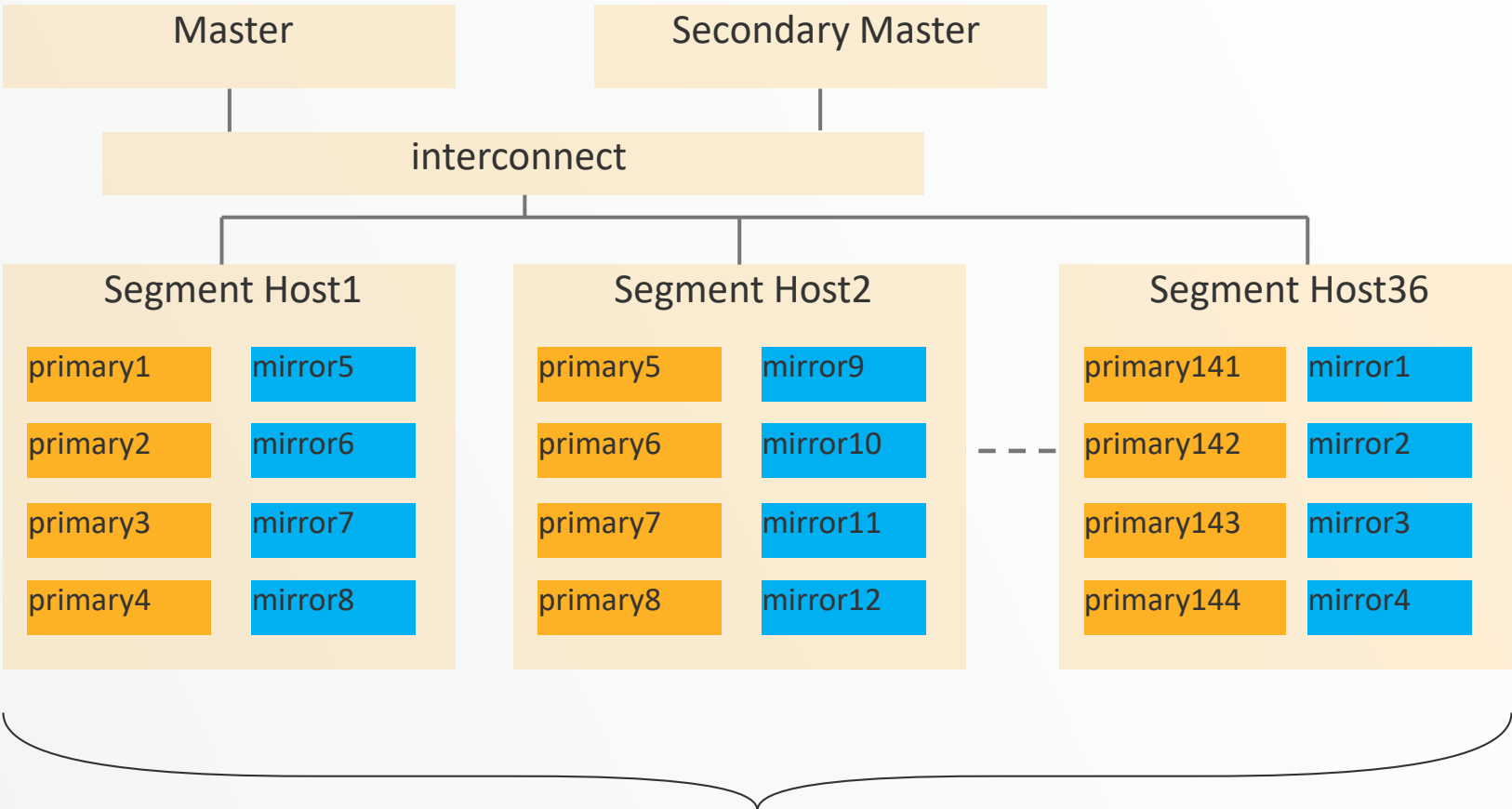
MPP кластер (9k cores; 3PB) * 10 шт.
25+ тыс аналитиков



Cloud compute + S3 storage
100k cores + 21 PB

Архитектура GP

- 1. GP - классическая **MPP** (massive parallel processing) архитектура
- 2. Сегменты (primary, mirror), независимые инстансы postgresql, mirror принимает запросы на себя при выходе мастера, RF=2
- 3. GP5 базируется на версии PostgreSQL 8.3, GP6 на PostgreSQL 9.4
- 4. Все данные (в идеале) равномерно распределены по всем сегментам
- 5. Запрос, обрабатывается **одновременно** на всех сегментах и работает со скоростью самого медленного
- 6. Все хосты стоят в одной зоне доступности



У нас либо **36** либо **72** ноды и 144 сегмента
это дает нам* **9К ядер** и **~3РВ данных** на кластер
**в пределе на текущем железе*

primary1

mirror5

— сегменты, инстансы postgresql

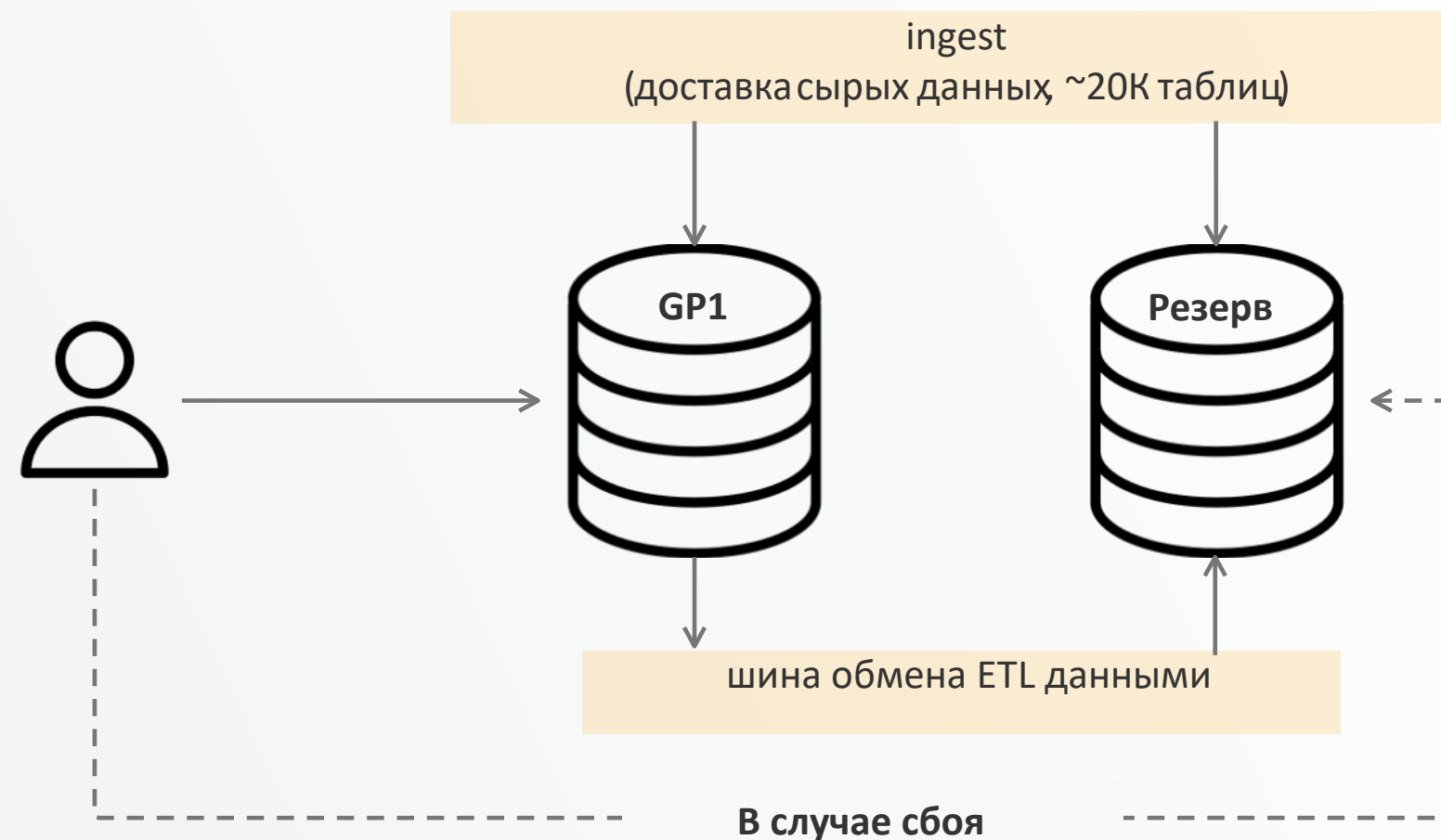
Segment Host1

— сервер

Надежность и хрупкость

1. Отказ любого сегмента приводит примерно к 20-30% потери производительности
2. Кластер не переживает отказ зоны доступности
3. За все годы мы так и не построили механизм горячей замены отказа мастера

Концепция – единого резерва

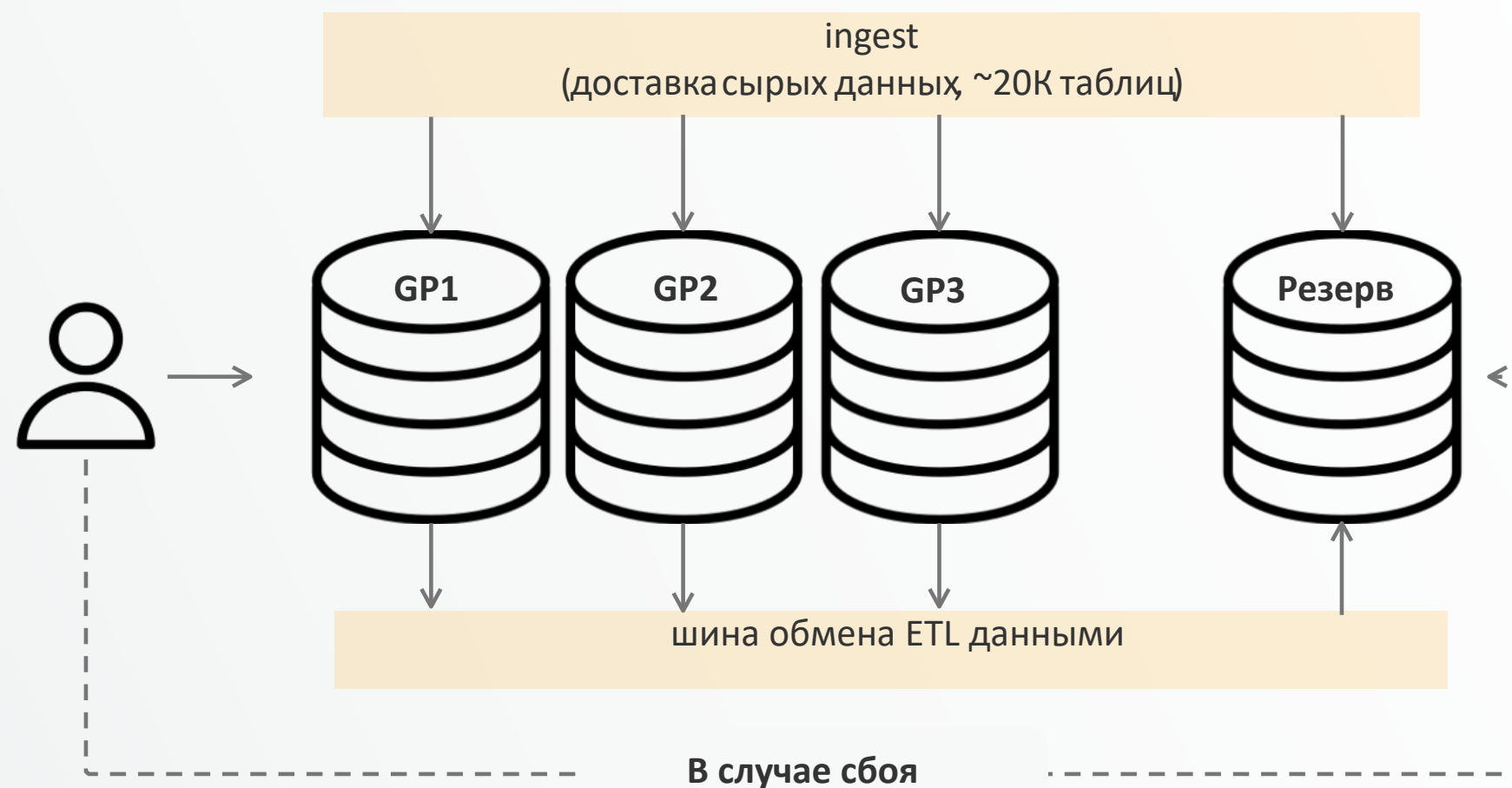


Все сырые данные доставляются как минимум на 2 кластера, для них нет бекапа в платформе, только на стороне баз источников
ETL данные распространяются на резерв через специальную шину, она же является для них бекапом
Резерв и GP1 стоят в разных зонах доступности
При сбоях везем пользователей на резерв

Масштабирование

1. Мы масштабируемся путем добавления новых кластеров
2. Мы не можем просто докинуть 10 или 20 серверов, только кластер
3. GP очень ограничен на количество параллельных сессий
4. Таблицы синхронизируются между кластерами, это порождает большое дублирование данных, но зато данные локальны
5. Мы постоянно решаем задачу “рюкзак” об эффективной рассадке всех наших пользователей по кластерам
6. Мы не можем заставить работать над задачей/запросом больше чем 72 сервера.
7. Шина обмена - еще одно узкое место, ее нужно масштабировать согласованно
8. Единый резерв должен вмещать данные всех кластеров

Наше масштабирование было сильно ограничено

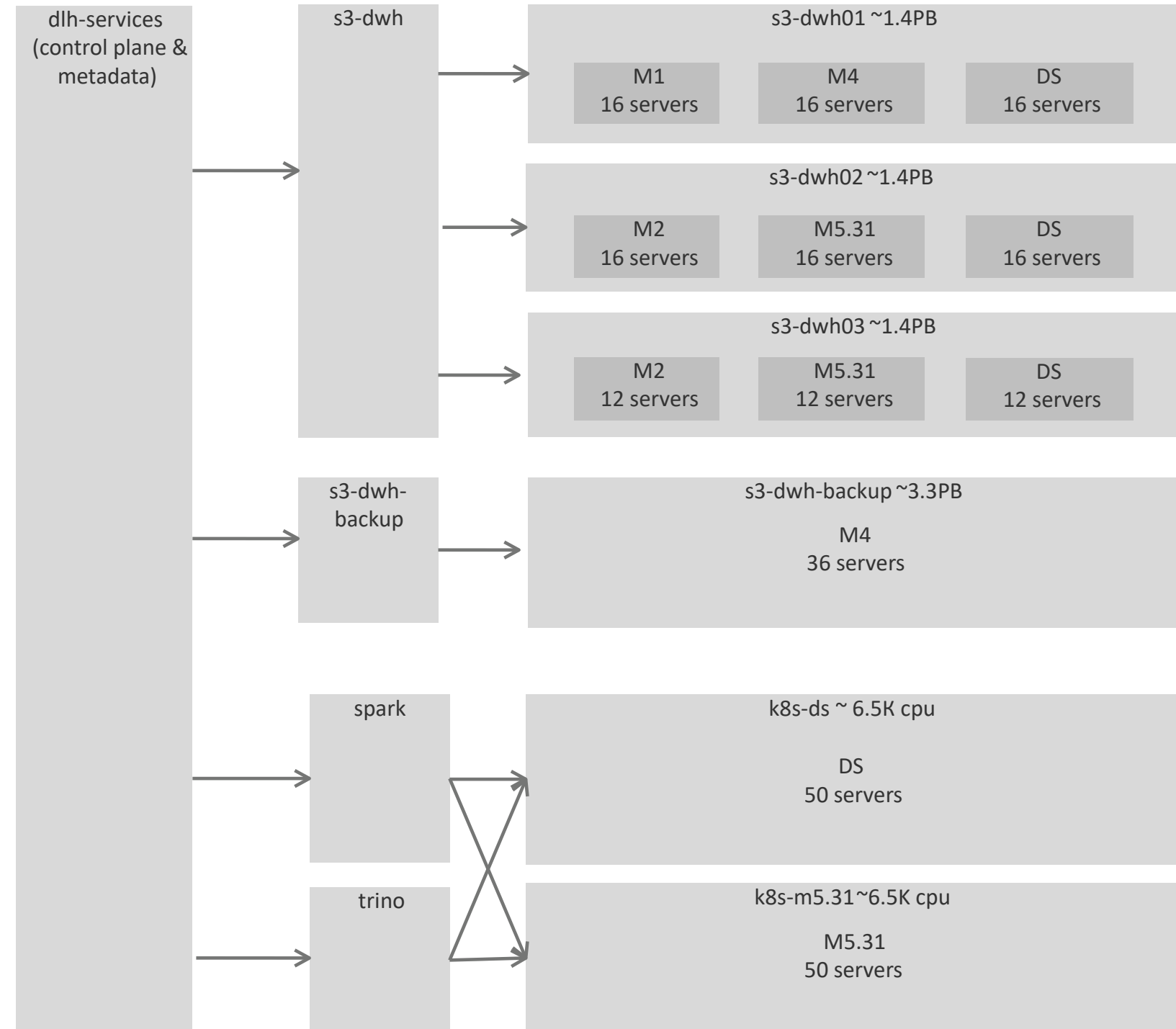


1. Мы можем масштабировать кластер в кол-ве серверов только если остается 144 сегмента (на этом базируется шина обмена)
2. Но 144 сервера уже считаем слишком хрупкой конфигурацией, поэтому больше 72 нод не собираем
3. Мы ограничены в закупке самых современных серверов

Распределенная Архитектура кластера Data Lake House (DLH)

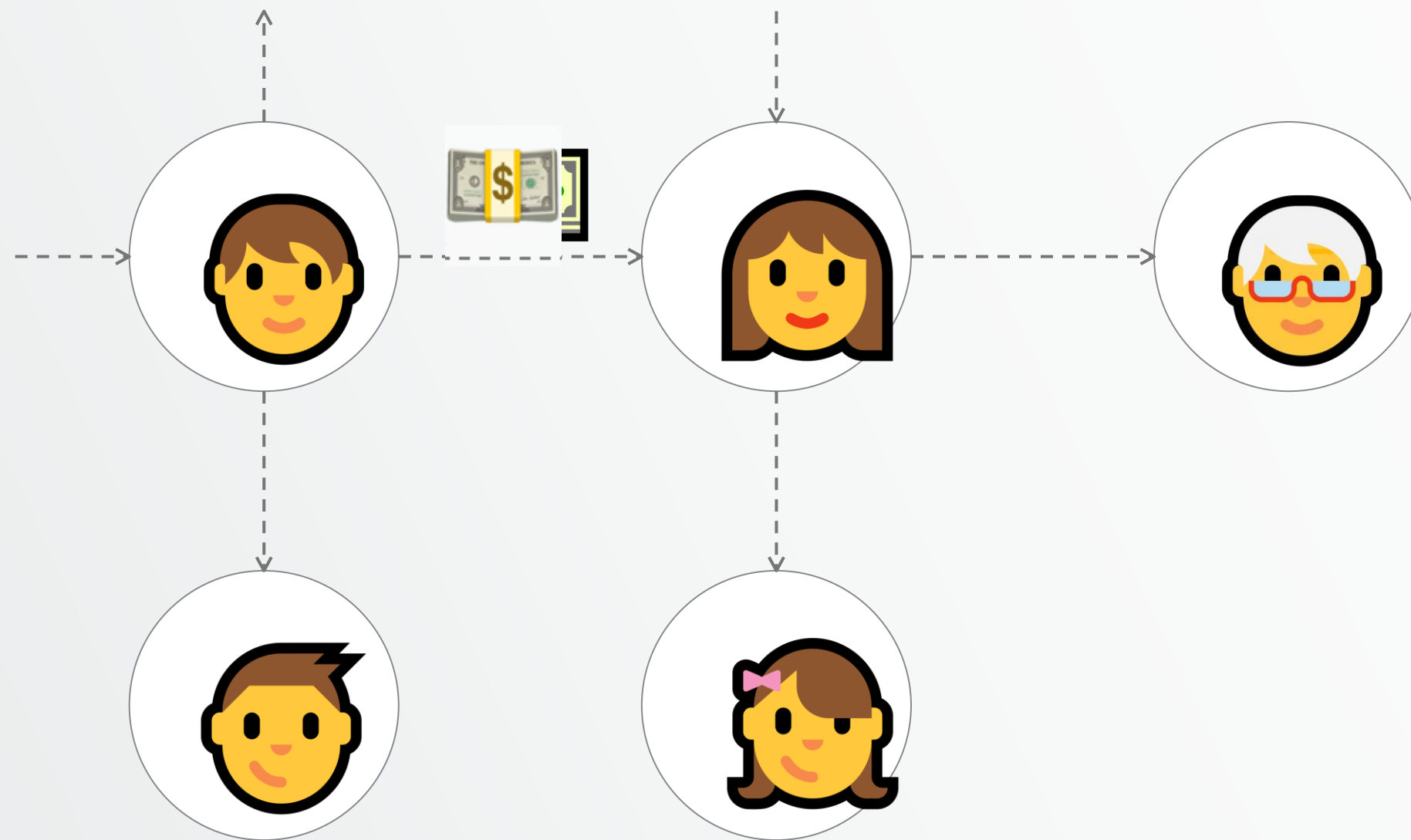
8+ тыс. аналитиков изучают данные

1. Storage (s3aaS)
2. Computeoverk8s
3. Data Platform отвечает за сервисы связывающие s3 и движки: метаданные, бекап, права доступа,etc
4. Важно строить достаточно большие «корзинки» так чтобы опять не столкнуться с проблемой «рюкзак»
5. До конца года вводим еще ~12PB под данные и 30PB под backup, 30K cpu для compute



Инструменты анализа больших графов

Инструменты анализа больших графов



В чем сложность?

Граф с которым нужно быстро работать

800 млн

вершин с атрибутами

80 млрд

связей между ними где тоже 20+ атрибутов на связь

Что такое ВАЖНЫЕ вершины и связи?

Центральности для вершин

Схожесть ребер

Принадлежность к группам

Близость к аномалиям в графе

...Как понять, что мы получили что-то ценное?

...и почти сразу упёрлись в стену вычислений в DWH,
а опенсорсные библиотеки не работают на масштабе

Удивительно что эти данные оказались ценными

Нужны инструменты для работы с большими графами

Команда R&D

Как сделать так, чтобы ценность из связей можно было реально быстро посчитать на графе в сотни миллионов вершин, строить и проверять гипотезы, получать данные для обучения и дифференса в проде.

01

Данные графа

Собираем и обновляем
большие графы
из данных компании

02

Инструменты

Считаем метрики,
кластеры и признаки
на больших графах

03

Ценность вроде

Отдаём результат модели
и продукты бизнеса

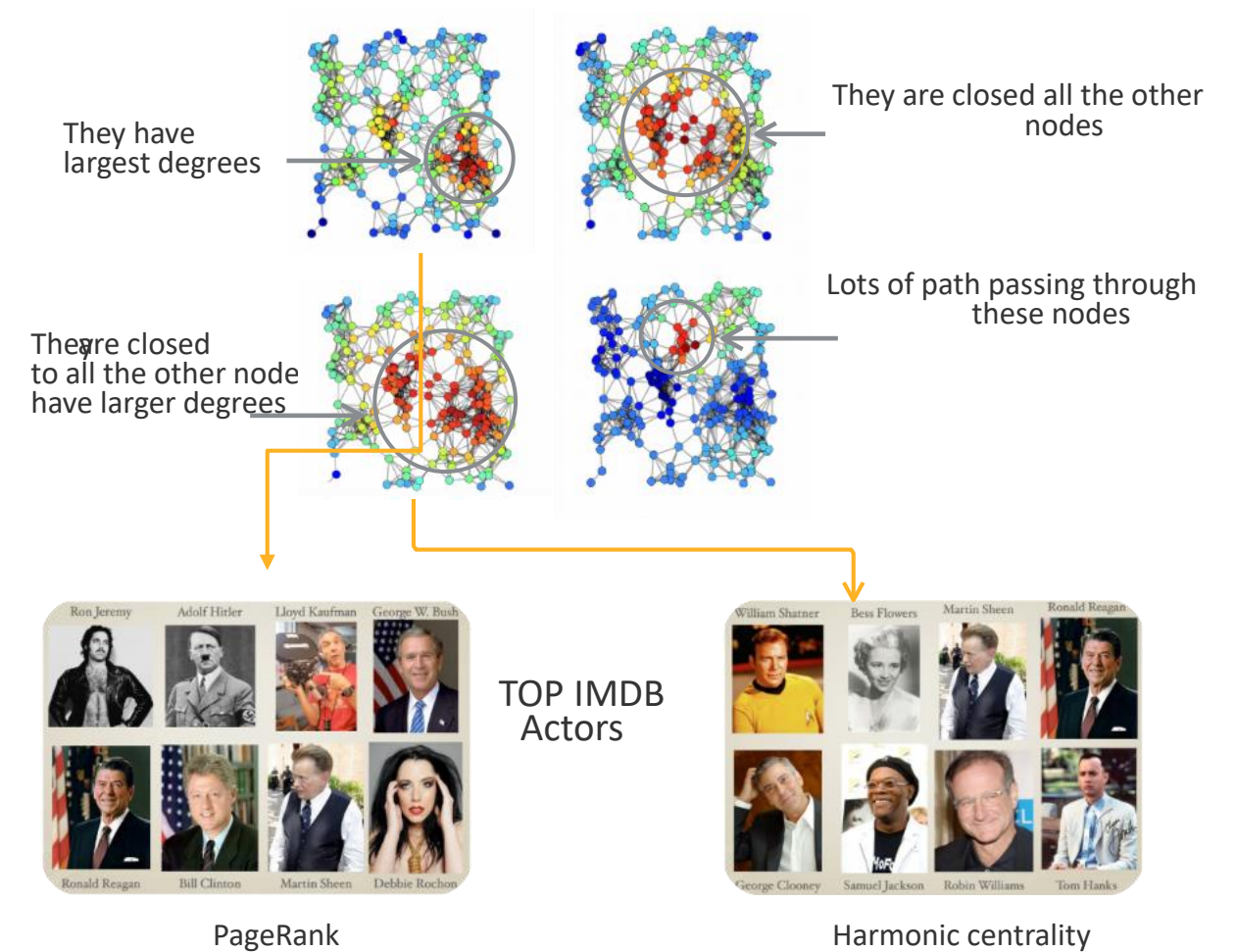
Почему аналитика на графе — задача для инженерного R&D

01 Методологическая сложность

02 Алгоритмическая сложность

03 Технологическая сложность

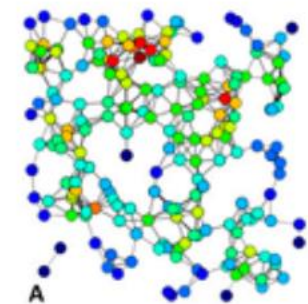
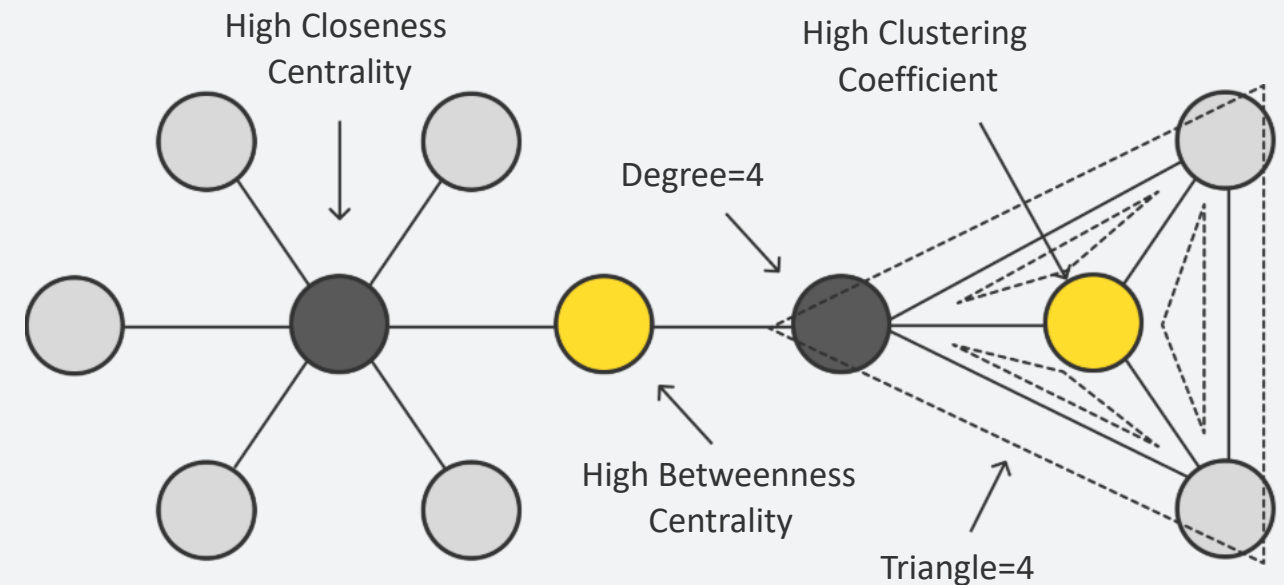
04 Большие размеры данных



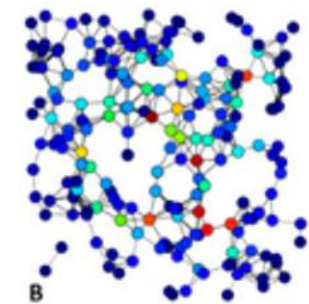
Как быстро понять чем ценен граф?

Необходимо:

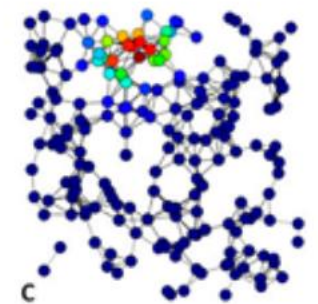
- Быстро и дешево предрасчитывать метрики на графе
- Эффективно хранить данные расчетов
- Исследовать запросами большой граф (искать соседей, атрибуты, обходить пути, считать агрегаты)
- Интерпретировать значения полученных метрик и определить ценность для разных задач
- Строить фичи для моделей с использованием графа, агрегировать с другими данными



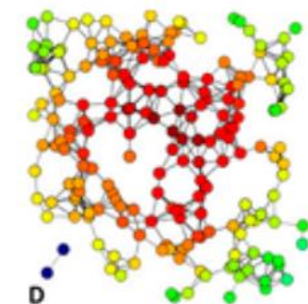
Наибольшая степень



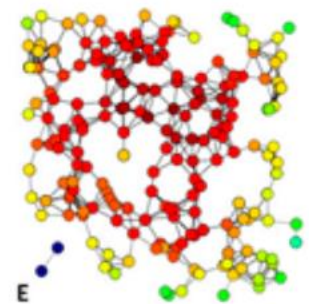
Узлы-концентраторы



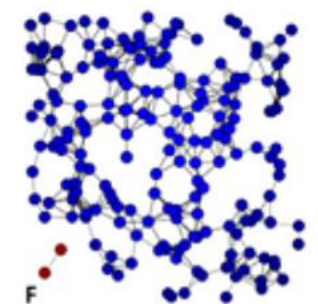
Влиятельные



Мосты



Мосты с высокой степенью



Мосты с нормализованной важностью

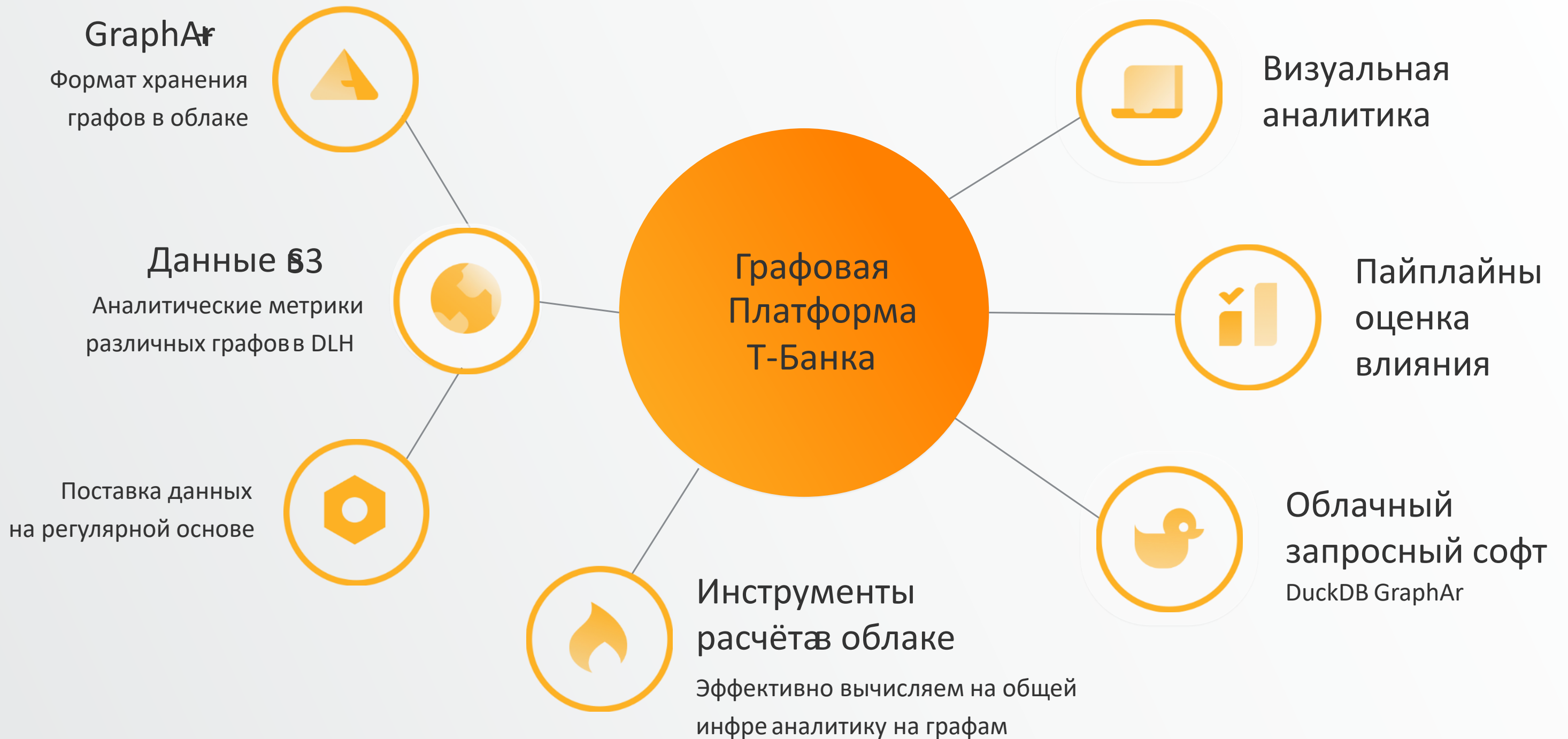
Где Т-Банк использует графы

Ценность графов для команд Т-Банка

- ➔ ML-модели с фичами вершин и связей из графа
- ➔ OLTP-сценариях анализа окружения клиента
- ➔ Расчеты на исторических данных
- ➔ Self-service-сценарии и discovery связей на графе
- ➔ Выявление сообществ и аномалий в графе связей



Графовая аналитика в облаке



Новые исследования на больших графах

Единая среда для работы с крупными графами дала возможность исследовать новые направления

GNN на графах

Доступ к специфическим шаблонам 2-hop окружения слишком медленный для реляционного хранилища.

GFM и RFM

Для обучения нужны большие объёмы графовых данных.

Фильтрация окружения

Как проредить окружение без потери качества предсказательных моделей на большом графе.

Статистические исследования

Проверка статистических гипотез требует инструментов исследования графа.

Исследование способов хранения данных в облаке

Labeled Property Graph (LPG) в табличной форме

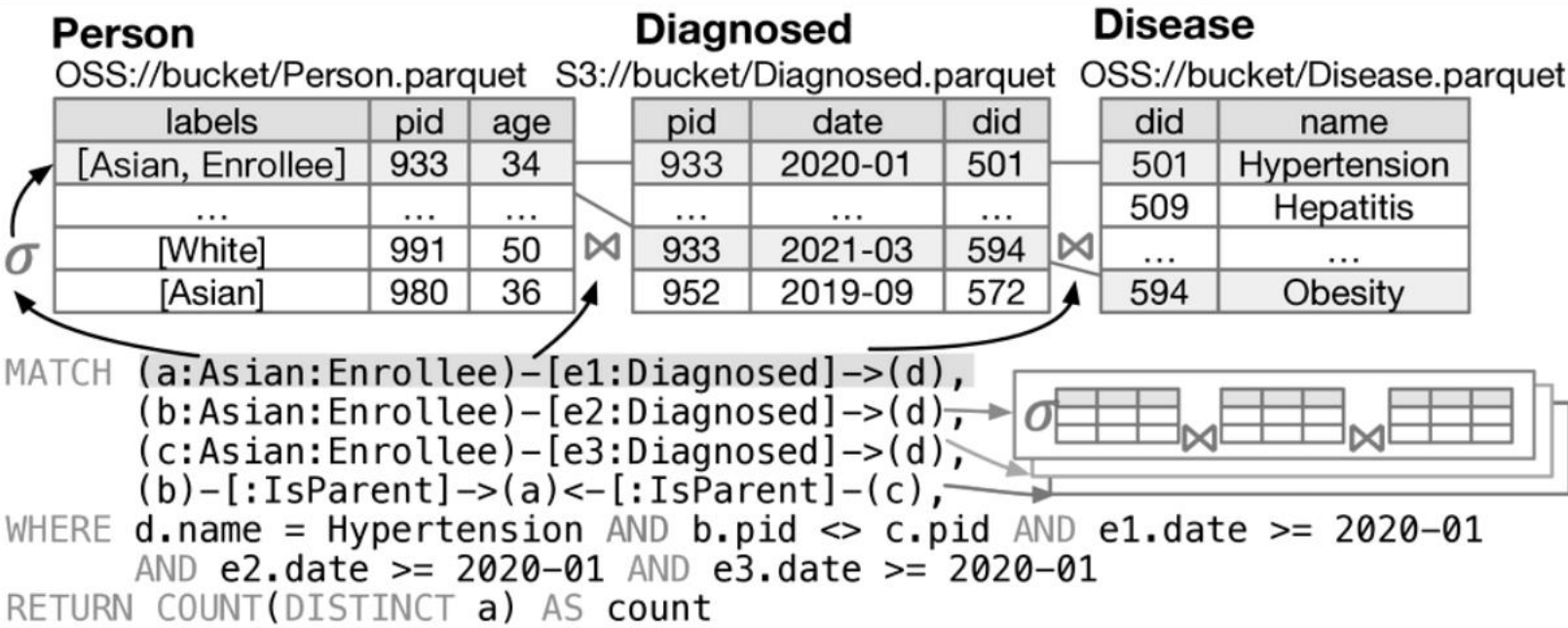
LPG это:

- вершины
- рёбра
- метки-классификаторы вершин
- свойства вершин и рёбер

Проблемы:

- нет универсального способа представить граф
- поиск соседей это куча JOIN-ов
- фильтрация меток не поддерживается

Пример процесса выполнения запроса LPG
в табличном представлении





Apache GraphAr: графовый архив

предложен в 2023 году компанией Alibaba

1 Открытый эффективный файловый формат для LPG в DLH

2 Совместим как с DLH, так и существующими инструментами и системами

3 Предлагает стандарт LPG для обмена данными между разнородными графовыми БД и системами



Apache GraphAr Home
<https://graphar.apache.org/>

Ограничения :

1 не заменяет существующие форматы в DLH, только добавляет функции для LPG

2 не вычислительный движок графов, просто архивный формат и источник данных – по сути графовый Storage

3 данные неизменяемы после их создания (но **можно добавить новые свойства** к существующим)



Apache GraphAr GitHub
<https://github.com/apache/incubator-graphar>

GraphAr как решение проблем LPG в DLH

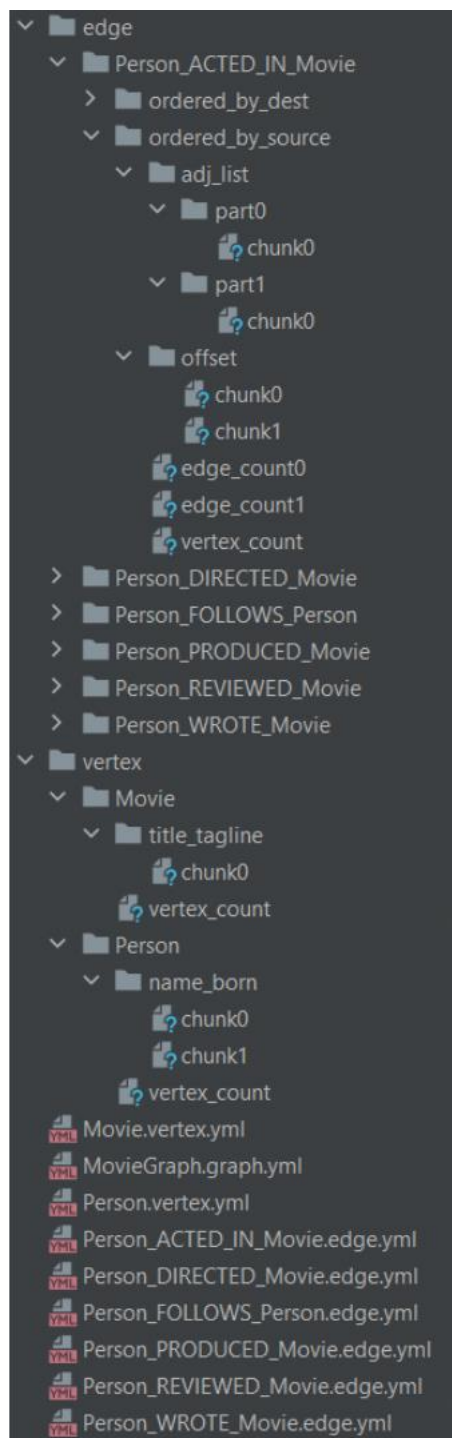
Сохранение семантики LPG и решение проблемы фильтрации меток

- файлы YAML для описания метаданных
- иерархическая структура файлов для контролируемой сегментации
- хранение данных в Parquet/ORC
- GraphAr адаптирует технику RLE (Run-Length Encoding) от Parquet и представляет новый алгоритм декодирования на основе интервалов

Решение проблемы поиска соседей

- Хранение рёбер в отсортированных таблицах Parquet в виде списков смежности:
 - order_by_source/ CSR (сжатая разреженная строка)
 - order_by_dest/ CSC (сжатый разреженный столбец)
 - unordered_by_source & unordered_by_dest/ COO (coordinate format), хранятся только ненулевые элементы и их номера строк и столбцов на основе группировки по внутреннему id
- Дельта-кодирование Parquet для снижения накладных расходов при хранении и загрузке данных
- Алгоритмы декодирования использующие векторизацию (MMI и SIMD)
- Выровненные по страницам коллекции (PAC) для дальнейшего ускорения процесса поиска соседей

Устройство GraphAr



```
MovieGraph.graph.yml
1 vertices:
2   - Person.vertex.yml
3   - Movie.vertex.yml
4   name: MovieGraph
5 edges:
6   - Person_DIRECTED_Movie.edge.yml
7   - Person_PRODUCED_Movie.edge.yml
8   - Person_REVIEWED_Movie.edge.yml
9   - Person_ACTED_IN_Movie.edge.yml
10  - Person_FOLLOWS_Person.edge.yml
11  - Person_WROTE_Movie.edge.yml
12 version: gar/v1
13
```

```
Person.vertex.yml
1 chunk_size: 100
2 prefix: vertex/Person/
3 property_groups:
4   - file_type: parquet
5     properties:
6       - is_primary: true
7         name: name
8         data_type: string
9       - is_primary: false
10        name: born
11        data_type: int64
12 type: Person
13 version: gar/v1
```

```
Person_REVIEWED_Movie.edge.yml
1 dst_chunk_size: 100
2 src_type: Person
3 chunk_size: 1024
4 src_chunk_size: 100
5 prefix: edge/Person_REVIEWED_Movie/
6 property_groups:
7   - file_type: parquet
8     properties:
9       - is_primary: false
10        name: rating
11        data_type: int64
12       - is_primary: false
13        name: summary
14        data_type: string
15 adj_lists:
16   - ordered: true
17     aligned_by: src
18     file_type: parquet
19   - ordered: true
20     aligned_by: dst
21     file_type: parquet
22 dst_type: Movie
23 edge_type: REVIEWED
24 version: gar/v1
```

- Доступ к данным без считывания всех данных в память
- Удобное добавление новых свойств к графу без необходимости реорганизации данных
- Эффективное хранение и доступ к данным в распределенной среде и параллельной обработке

Инструменты доступа к данным

DuckDB extensions

DuckPGQ

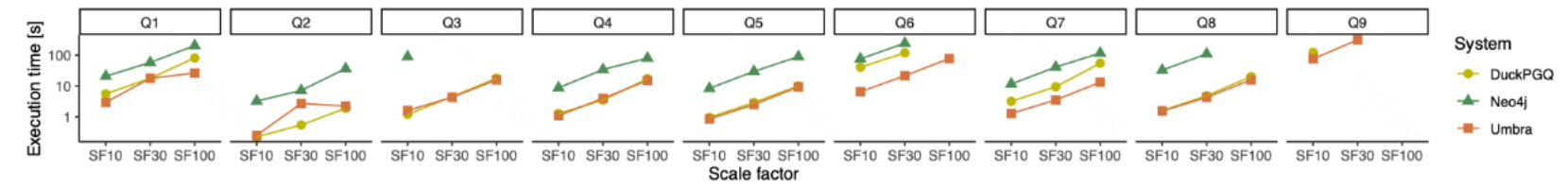
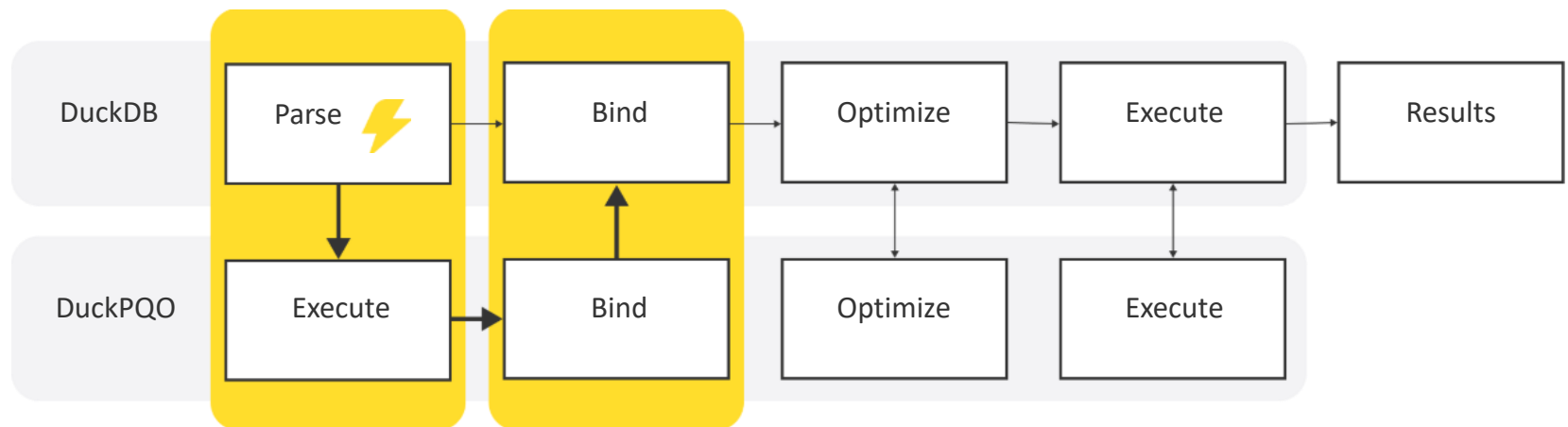


Figure 2: Pattern matching performance on the LSQB

DuckPGQ: Efficient Property Graph Queries in an analytical RDBMS

Daniel ten Wolde CWI The Netherlands dljtw@cwi.nl	Tavneet Singh CWI The Netherlands tavneet.singh@cwi.nl	Gábor Szárnyas CWI The Netherlands gabor.szarnyas@cwi.nl	Peter Boncz CWI The Netherlands boncz@cwi.nl
--	---	---	---

SQL/
PGQ

```
SELECT count(id) AS cp2
FROM GRAPH_TABLE (socialNetwork,
MATCH (p1:person WHERE p1.name='Bob')-[:follows]->*(p2:person)
-[:livesIn]->(c:city WHERE c.name='Utrecht'))
COLUMNS (p2.id))
```

plain
SQL

```
WITH RECURSIVE paths(startNode, endNode, path) AS (
SELECT p1id AS startNode, p2id AS endNode, ARRAY[p1id, p2id] AS path
FROM follows JOIN person p1 ON p1.id = follows.p1id WHERE p1.name = 'Bob'
UNION ALL (
WITH paths AS (TABLE paths)
SELECT paths.startNode AS startNode, p2id AS endNode, array_append(path, p2id) AS path
FROM paths JOIN follows ON paths.endNode = follows.p1id
WHERE NOT EXISTS (SELECT true FROM paths previous_paths
JOIN person p2 ON p2.id = follows.p2id
WHERE p2.name = 'Bob' OR follows.p2id = previous_paths.endNode)))
SELECT count(p2.id) AS cp2
FROM person p1
JOIN paths ON paths.startNode = p1.id
JOIN person p2 ON p2.id = paths.endNode
JOIN city ON city.id = p2.livesIn AND city.name = 'Utrecht'
```

The SQL/PGQ query is 4× more concise

SF	Path-finding		LSQB	
	Pers.	knows	V	E
10	70k	2M	35M	217M
30	175k	6M	103M	650M
100	487k	23M	326M	2B
300	1M	68M	n/a	n/a
1 000	3M	227M	n/a	n/a
3 000	9M	670M	n/a	n/a

Table 1: Statistics of the graphs

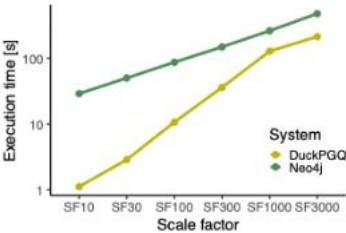


Figure 3: Path-finding performance

Duck-GraphAr

→ Расширение duck-graphar для DuckDB, добавляет read-only поддержку формата GraphAr

→ Поддержка диалекта SQL/PGQ и его оптимизаций

→ Работает напрямую S3 в zero-copy режиме

→ Измерения получившегося решения в сравнении с другими форматами в рамках экосистемы DuckDB показывают значительный эффект в скорости и потреблении ресурсов



→ Расширение опубликовано как OpenSource

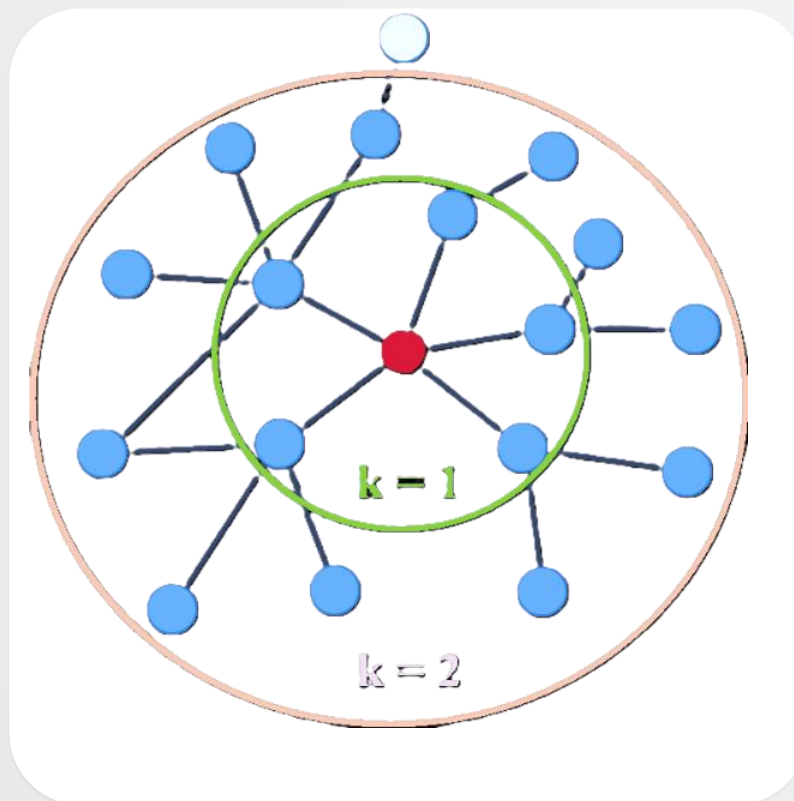


Сценарии семплинга

1-hop

Ближайшее окружение вершины

Актуально для графовых свёрток (скор по усреднённому окружению)



2-hop

Друзья-друзей вершины

Актуально для:

- GNN - Graph Neural Network
- сэмплирования графа
- анализа пересечения окружений

Дорого для больших графов в табличном представлении

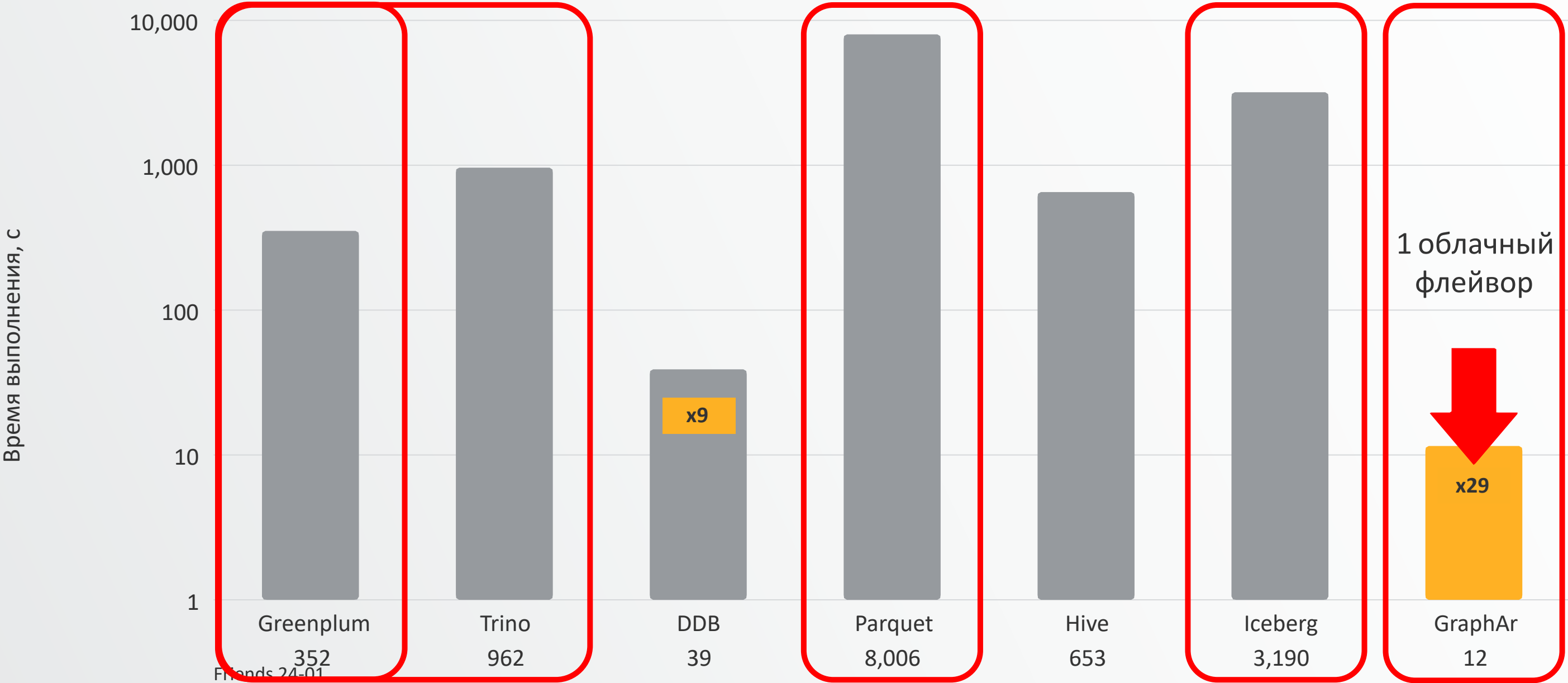
Поиск связи

Проверить наличие связи между двумя вершинами

Дорого
для больших графов

Очень-очень дорого
для больших графов
в табличном представлении

duck-graphar: поиск 2-hop соседей



Кластер платформы данных

Средний размер 2hop для одной вершины 5,5 миллионов

DuckDB DuckPGQ Duck- GraphAr

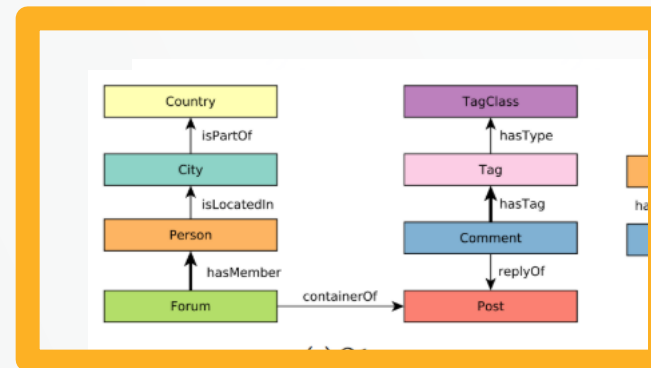
Запросный сетеп для больших графов

Показывает высокую производительность для больших графов на бенчмарках LDBC SNB и LSQB

Имеет оптимизированный функционал для семплинга большого графа и подготовки данных для ML в виде UDF для вариаций 1-hop и 2-hop

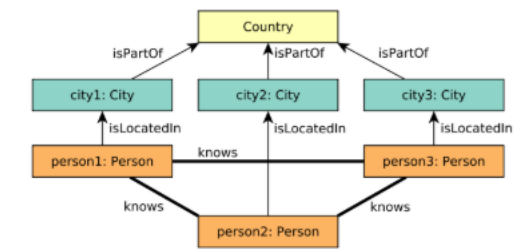
Идет работа над оптимизацией плана запроса над GraphAr

Доступен аналитикам в базовом образе графовой платформы



```
1 SELECT count(*)
2 FROM Country
3 JOIN City
4   ON City.isPartOf_CountryId = Country.CountryId
5 JOIN Person
6   ON Person.isLocatedIn_CityId = City.CityId
7 JOIN Forum_hasMember_Person
8   ON Forum_hasMember_Person.PersonId = Person.PersonId
9 JOIN Forum
10  ON Forum.ForumId = Forum_hasMember_Person.ForumId
11 JOIN Post
12  ON Post.Forum_containerOfId = Forum.ForumId
13 JOIN Comment
14  ON Comment.replyOf_PostId = Post.PostId
15 JOIN Comment_hasTag_Tag
16  ON Comment_hasTag_Tag.CommentId = Comment.CommentId
17 JOIN Tag
18  ON Tag.TagId = Comment_hasTag_Tag.TagId
19 JOIN TagClass
20  ON Tag.hasType_TagClassId = TagClass.TagClassId;
```

(b) Q2.



(c) Q3.



(f) Q6.



(i) Q9.

ss SF10, SF30, and SF100

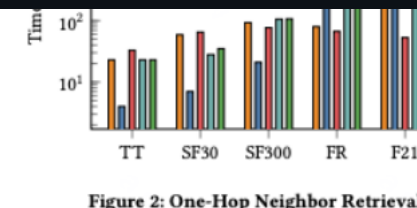
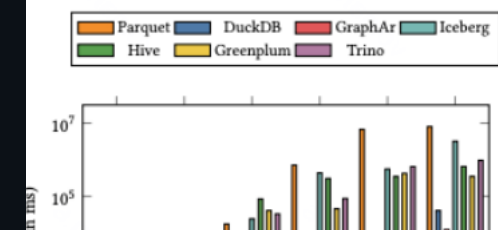


Figure 2: One-Hop Neighbor Retrieval

```
1 SELECT count(*) FROM GRAPH_TABLE (lsqb
2   MATCH (c:Country)<-[ipo:City_isPartOf_Country]-(c1:City)
3   <-[ilo:Person_isLocatedIn]-(p:Person)
4   <-[hm:hasMember]-(f:Forum)-[co:containerOf]->(po:Post)
5   <-[ro:replyOf_Post]-(c2:Comment)-[ht:Comment_HasTag]
6   ->(t:Tag)-[ht2:hasType]->(tc:TagClass)
7 );
```

Запросы к GraphAr

```
ATTACH '/friends/2021/FriendsGraph.yaml' AS friends2021 (TYPE duckdb_graphar);

ATTACH 's3://rnd-grapher-graph-output/graphar/friends/lpg/21-01-01/FriendsGraph.yaml' AS friends2021s3 (TYPE duckdb_graphar);

SELECT COUNT(*) FROM person;

SELECT COUNT(*) FROM person_knows_person WHERE _graphArSrcIndex=698456539;

CREATE PROPERTY GRAPH friends2021
VERTEX TABLES (
    person
)
EDGE TABLES (
    person_knows_person SOURCE KEY (_graphArSrcIndex) REFERENCES person (_graphArVertexIndex)
                        DESTINATION KEY (_graphArDstIndex) REFERENCES person (_graphArVertexIndex)
                        LABEL knows
);

SELECT count(*) FROM GRAPH_TABLE (friends2021
    MATCH (a:person WHERE a._graphArVertexIndex = 588636898)-[k1:knows]->(b:person)
);

SELECT * FROM two_hop('/friends/2021/FriendsGraph.yaml', vid={vid});
```

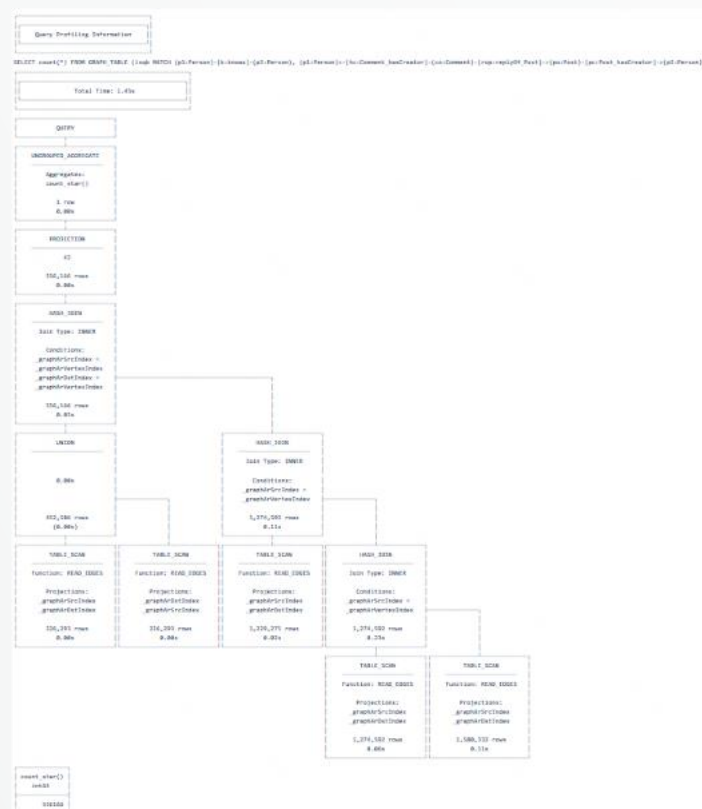
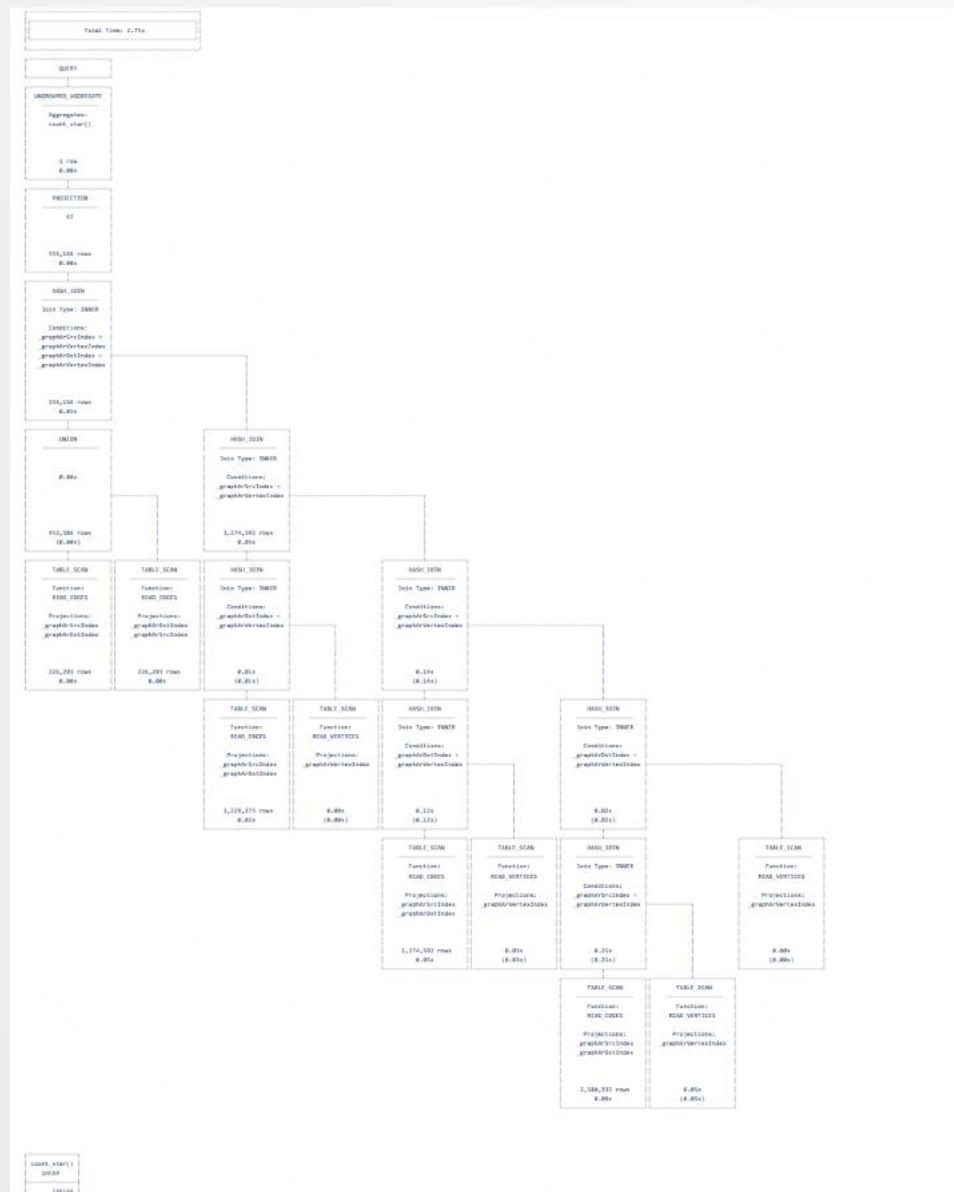
Оптимизация планов исполнения (join → udf)

```
SELECT count(*)  
  FROM GRAPH_TABLE (lsqb  
    MATCH  
      (p1:Person)<-[hc:Comment_hasCreator]-(co:Comment)-[rop:replyOf_Post]->(po:Post)  
    );
```

LSQB Q2 SF100

199 sec → 22 sec

```
SELECT count(*)  
  FROM GRAPH_TABLE (lsqb  
    MATCH (p1:Person)-[k:knows]-(p2:Person),  
      (p1:Person)<-[hc:Comment_hasCreator]-(co:Comment)-[rop:replyOf_Post]->(po:Post)-[pc:Post_hasCreator]->(p2:Person)  
    );
```



Карта компонент графовой платформы в Т

Данные	Графы предметных областей	Конвейер внесения нового графа	Контекстные графы	Графы по коду
Хранение	Специализированное хранилище на доработанном формате GraphAr		OLTP-решение (СУБД)	
Вычисление	ga — расчётная утилита	pyga — Python-интерфейс	Графовый бэкенд данных для фундаментальных моделей (GFM/RFM)	
	Темпоральные метрики и эмбединги	Шаблоны доступа к окружениям для GNN	Инфраструктура обучения графовых решателей (GNN)	
Доступ	Запросный движок DuckDB + расширение GraphAr	Интеграция инструментов в Helicopter и MLC	DuckDB as a Service	Поставка в FeatureStore/ DLH
Потребление	Визуальная аналитика	Streaming	AI Агенты	Базис построения фичей
Контроль	Качество данных	SLA	MAU	Ресурсы
Процесс	Продуктовая поддержка продавне R&D (Data Monetization)		self-service конфигурации	

Алгоритмические исследования

Оптимизация логистики

Платформа доставок

- Обеспечивает доставку продуктов Т-банка клиентам
- Более 80 регионов России

Представителей

11к

5к - 6к каждый день

Встреч

2 млн

В месяц по более чем
25 продуктам

Кост

1.5 млрд

В месяц. 80% -
вознаграждение
представителей

Как масштабировать
экономии на каждом
маршруте на всю
страну?

Оптимизация логистики

Иерархическая кластеризация

Делим карту на компактные районы при помощи алгоритма Лейдена рекурсивно

Оценка маршрута

Считаем расстояние между заказами алгоритмом Кристофидеса-Сердюкова

Рабочие дни региона

Выбираем оптимальные рабочие дни регионы при помощи Integer Linear Programming

Распределение представителей

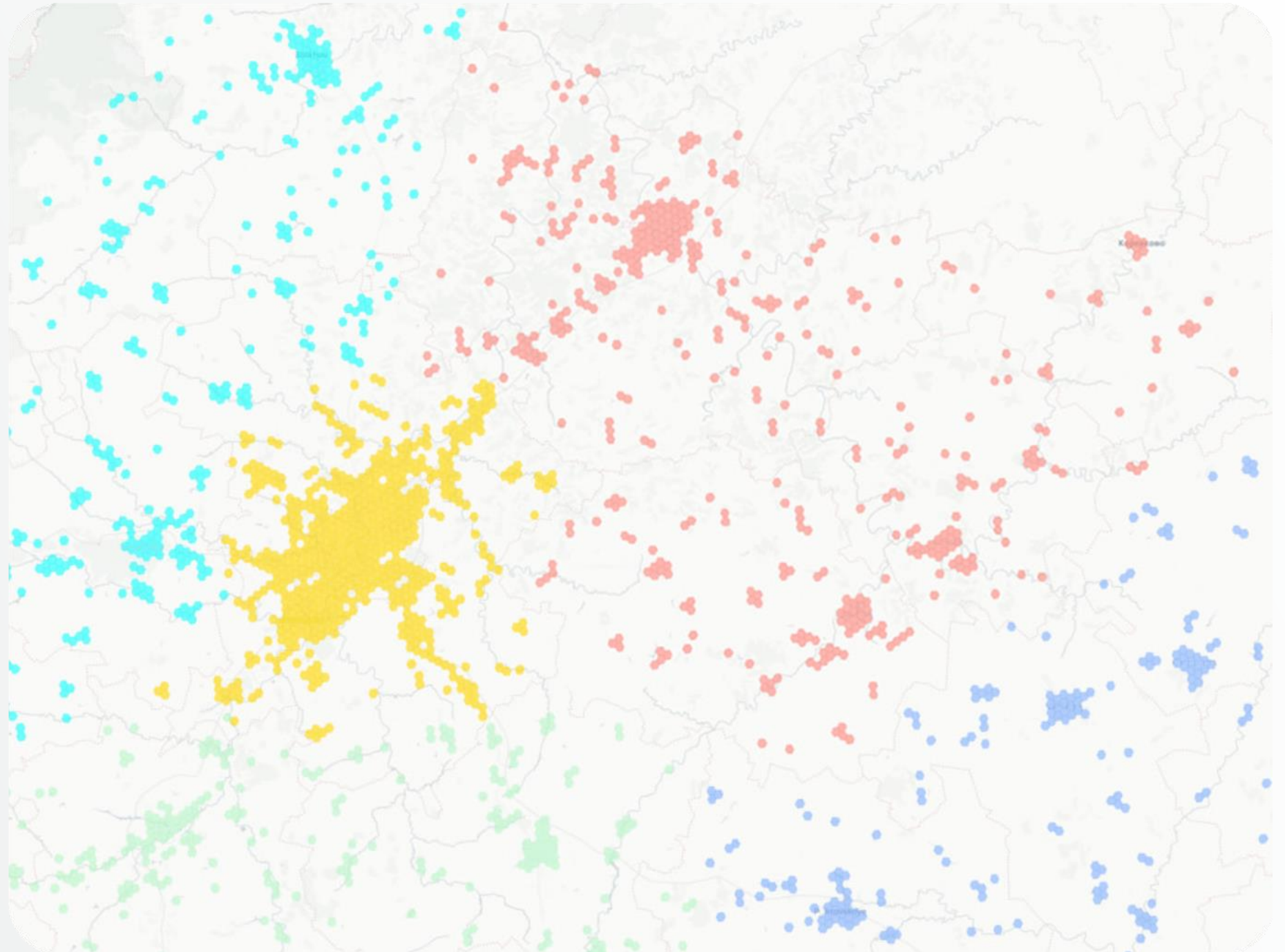
Назначаем представителей по регионам при помощи потоковых алгоритмов на графах

Режем карту на «кучные» районы

Чем плотнее заказы в районе —
тем быстрее и дешевле доставка.

Кластеризуем гексагоны карты (алгоритм
Лейдена) в компактные зоны доставки.

Каждый цвет — отдельный район одного
представителя.



Иерархическая кластеризация

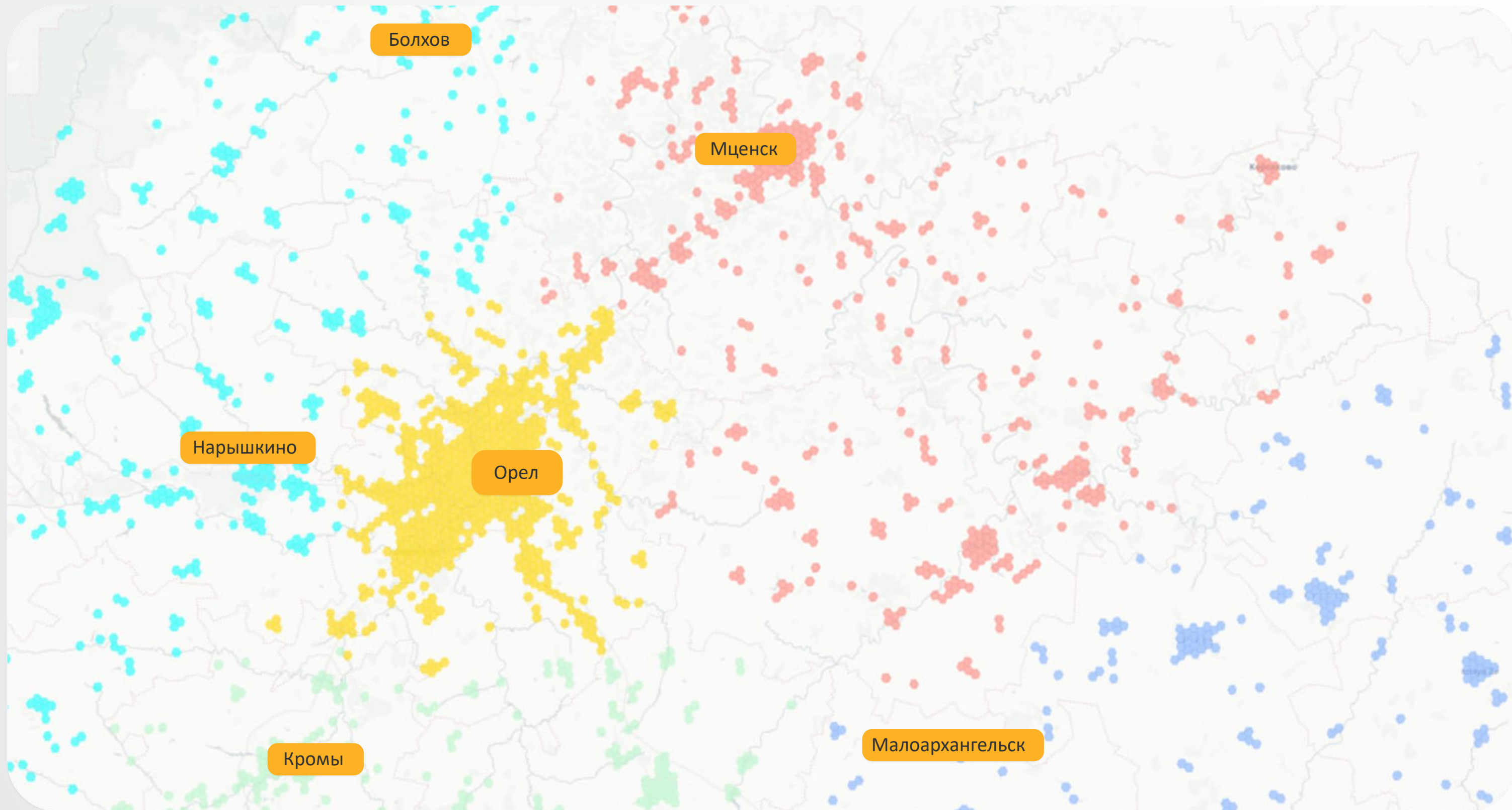
Как мы делаем :

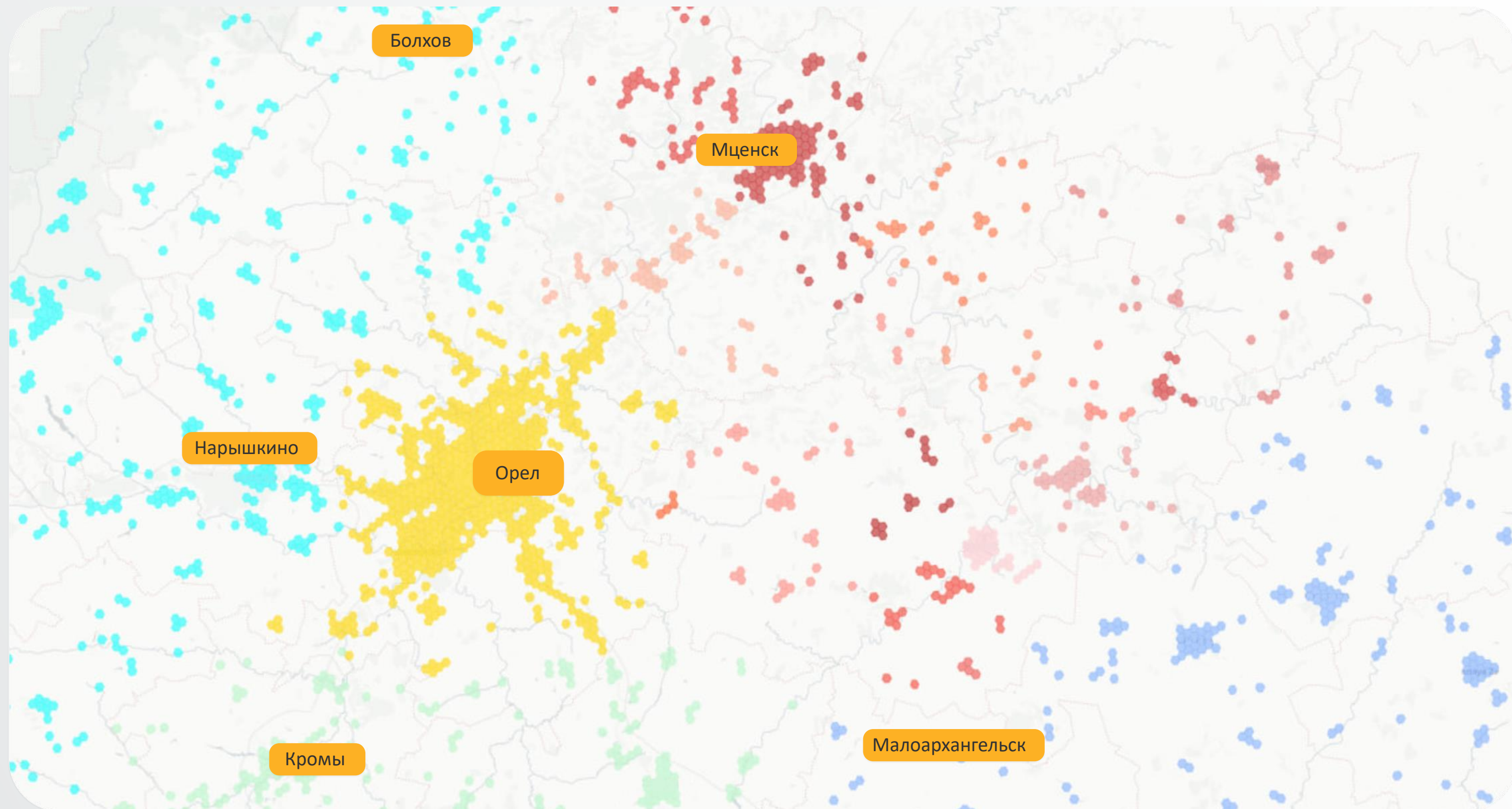
- ➔ делим всю карту России на крупные районы
- ➔ проверяем каждый район если выгодно – делим его на более мелкие кучные подрайоны

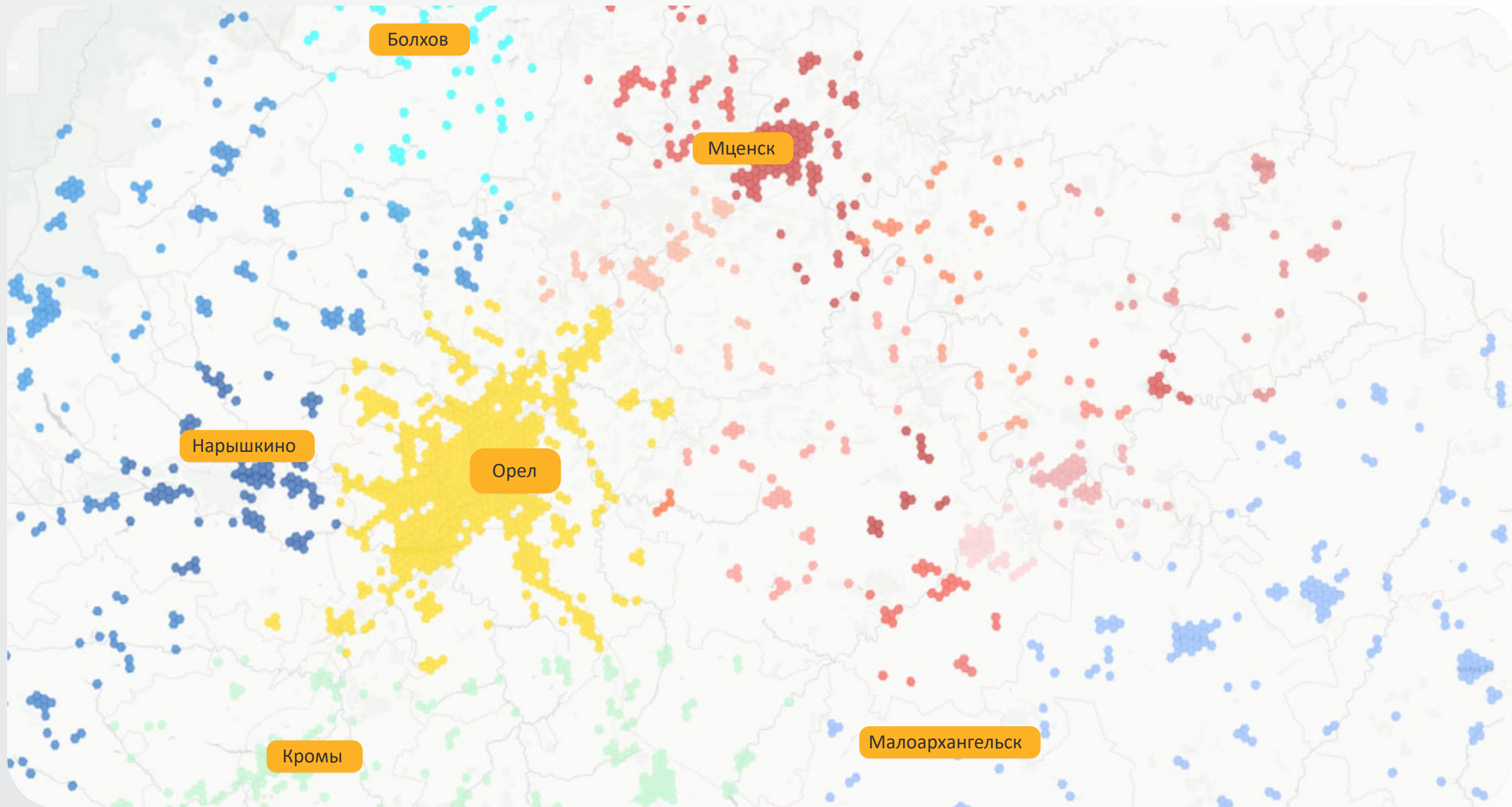
Зачем:

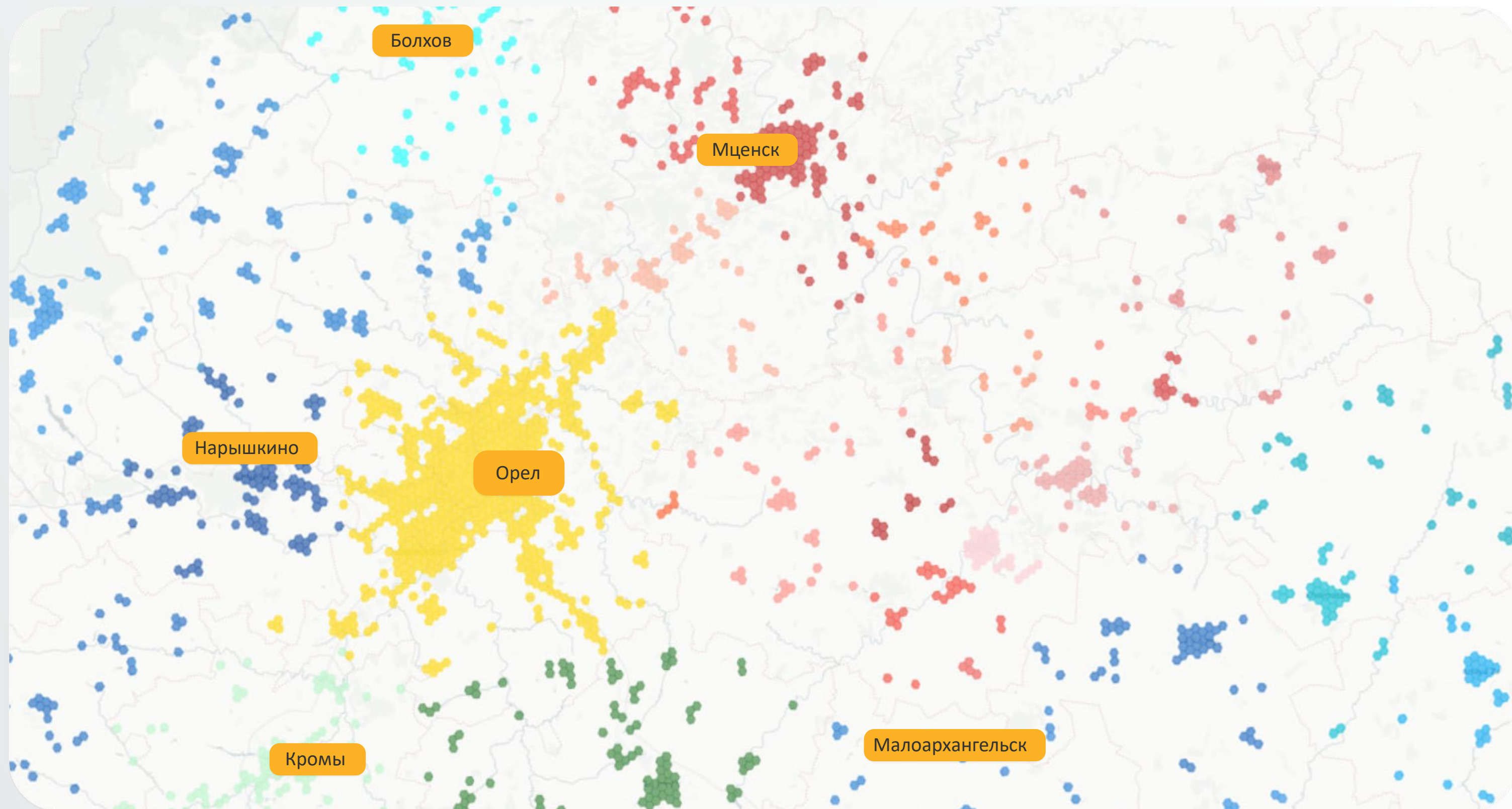
- ➔ строим районы с кучными заказами
- ➔ выделяем крупные населенные пункты и «плотные» районы внутри них
- ➔ отделяем малонаселенные деревни от «плотных» городских зон

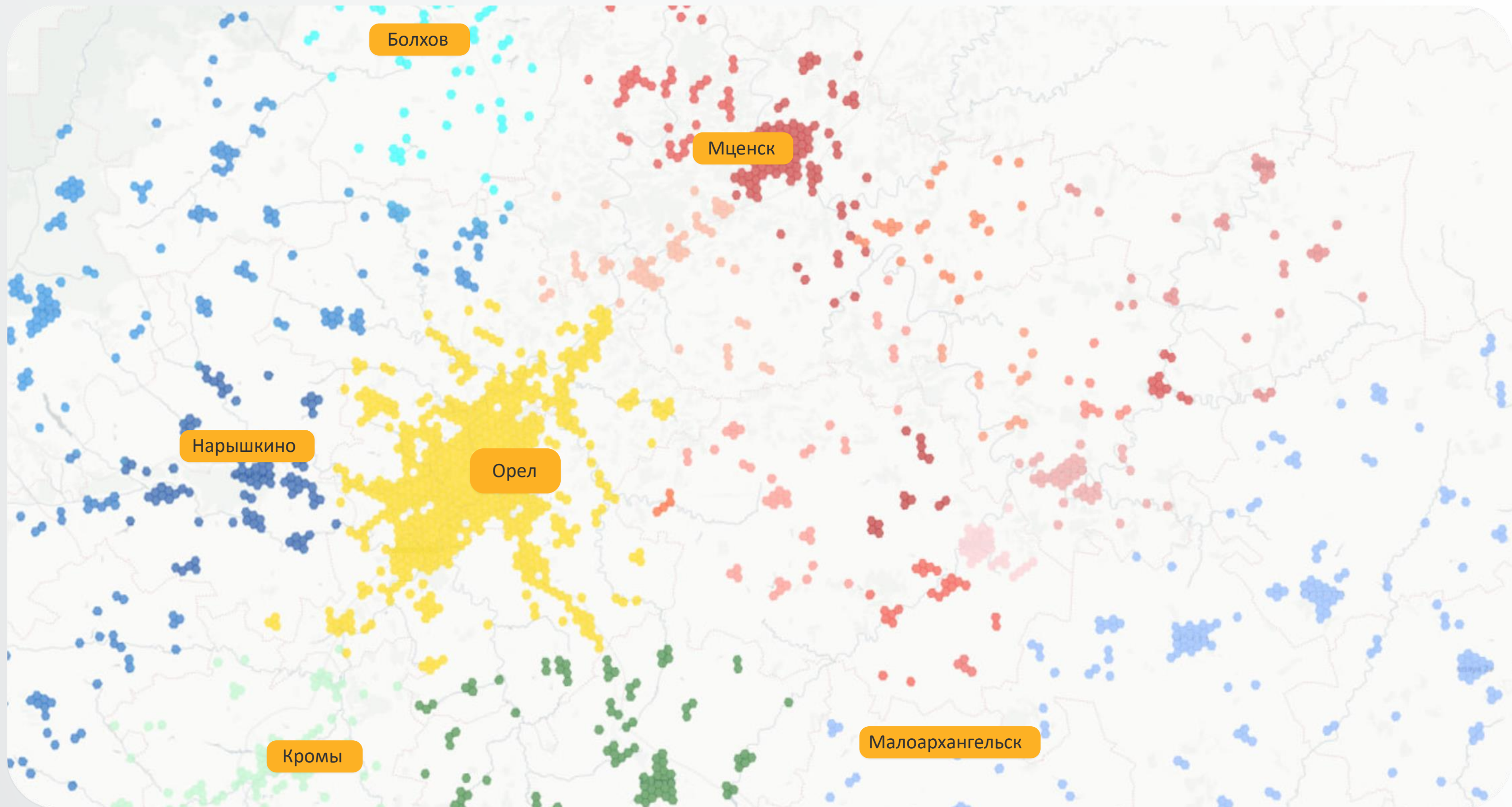












Больше заказов — меньше километража

Орловская обл.

+3.3%
заказов

−2.4%
километража

Ингушетия

+3.2%
заказов

−27.1%
километража

Еврейская АО

+20.4%
заказов

−6.5%
километража

Адыгея

+15.6%
заказов

−2.5%
километража

Экономия — миллионы ₽ в год. Пересчёт под акции и сезон — за минуты.

Оптимизация ЛОГИСТИКИ

Статус внедрения

Используется в 16 регионах России
Продолжается масштабирование на всю Россию

Средний километраж на заказ :
5.07 км → 4.6 км
-9.3%

Доставленных заказов :
+ 100 000 заказов в год
+ 1.15%

Результаты внедрения

Было	Стало
Расчет по всей стране занимал до недели	Расчет по всей стране за несколько часов
Экспертные эвристики	Формализованная математическая модель
Неточности в оценки километража между заказами	Точное оценка километража между заказами и прогноз временных окон региона
Ручные проектирование рабочих дней территориальными менеджерами	Автоматический и оптимальный выбор рабочих дней в регионах

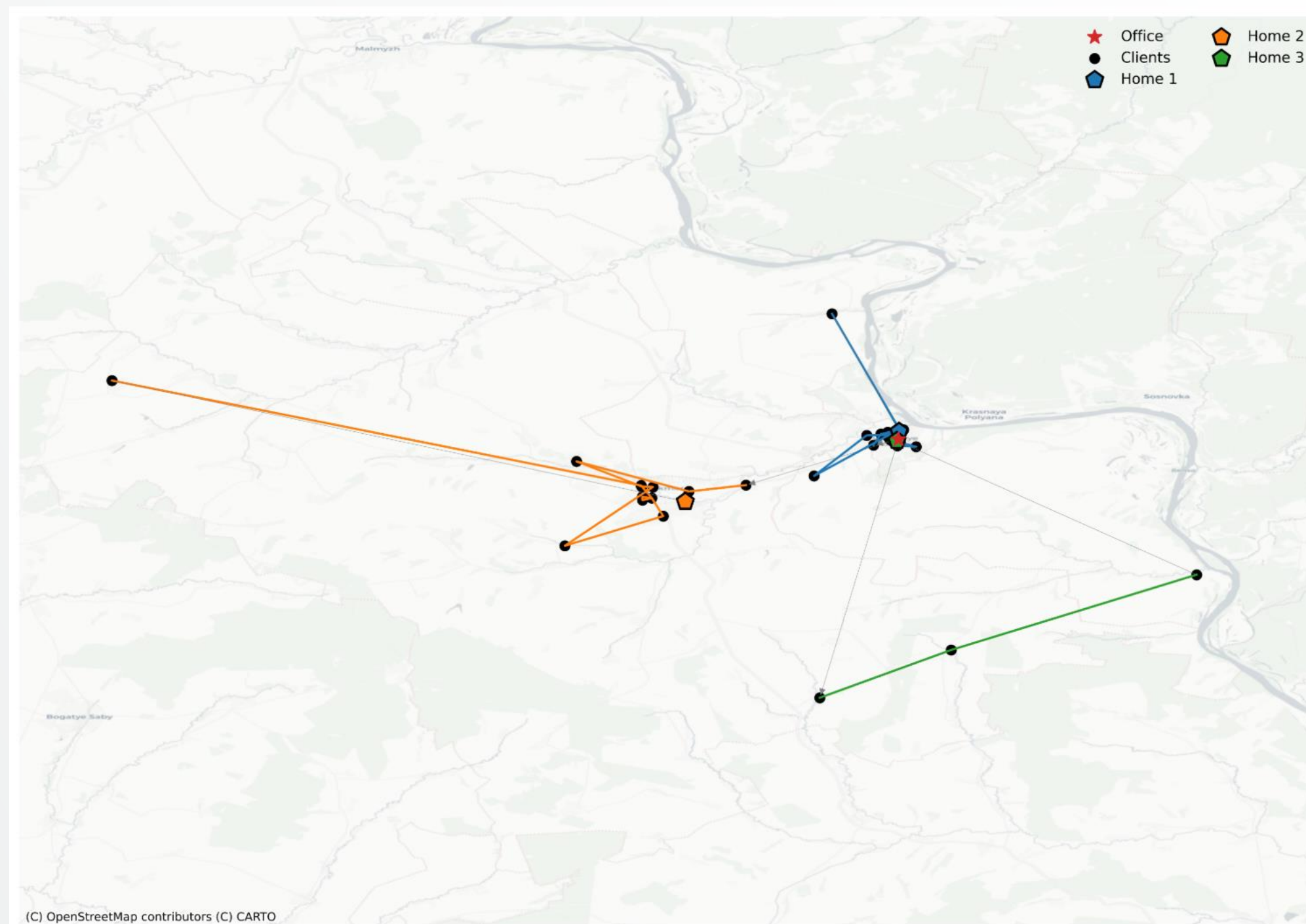
Оптимизация маршрутов представителей

Постановка задачи:

- Распределить заказы в каждом кластере между представителями
- Максимизировать соответствие компетенций представителя встрече
- Минимизировать затраты на километраж и опоздания

Как решаем:

- SOTA-реализация ЛАНС эвристики для задач логистики на C++
- Гибкий фреймворк для имплементации произвольных бизнес-ограничений



Предварительные результаты

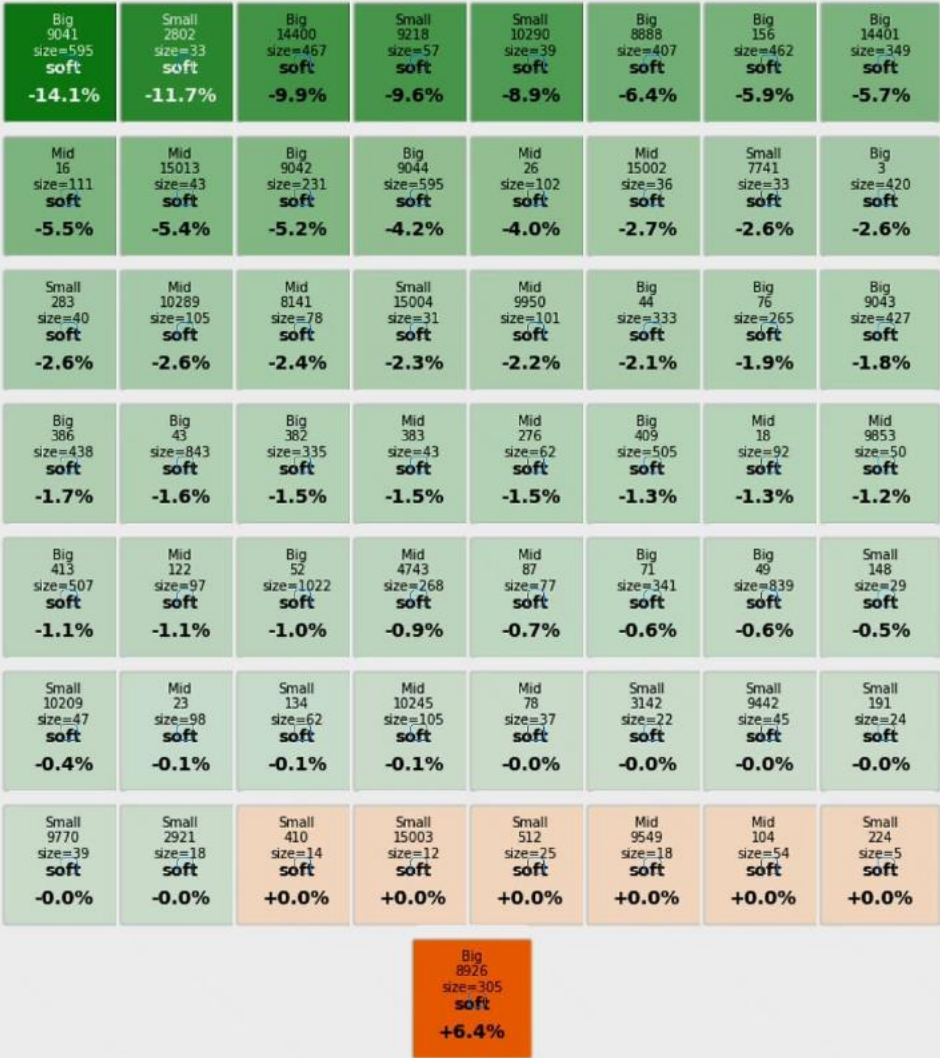
- 57 регионов
- 11,838 клиентов

$$Cost(r) = \sum_{i \in Hard} \Omega_i h_i(r) + \sum_{j \in Soft} w_j s_j(r) \rightarrow \min,$$

Критические ограничения
(соответствие компетенций
и опыта)

Мягкие бизнес-метрики
(километраж, опоздания)

-3.0% по мягким метрикам



Timefold (предыдущее решение)	PyVRP + фреймворк для ограничений
Реализация на Java:	Реализация на C++
гибкая, но не производительная	на основе SOTA солвера с расширением на кастомные бизнес-метрики
Универсальная специализация:	Узкая специализация:
Движок не адаптирован под VRP-структуру	Движок использует подход с VRP-сегментами для проверки движений за O(1)

Вызовы, требующие R&D и научных подходов

Источники вызовов R&D

Рост масштабов компании и инфраструктуры

Новые ЦОДы

Рост compute и storage инфры

Переход на IPv6

Искусственный интеллект и его эффекты в бизнесе

Исследование новых архитектур моделей

AI4SDLC

Инфраструктура compute и Storage для AI

Повышение надежности и эффективности решений

Снижение влияния сбоев

Утилизация инфраструктуры

Новые направления для бизнеса

Blockchain и криптовалюты

Робототехника

И многие другие.

Blockchain R&D

AML и On-chain аналитика

Собственная AML-платформа, скоринг адресов, мониторинг транзакций и аудит смарт-контрактов

Перспективная криптография

ZKP, MPC, FHE и постквантовые алгоритмы для защиты данных и вычислений

Приватность в публичных сетях

Обеспечение банковской тайны при работе с публичными блокчейнами, сетевая конфиденциальность

DeFi и инвестиционные стратегии

Анализ и разработка Earn-стратегий в DeFi-протоколах, оптимизация доходности

Blockchain R&D

2/2

Децентрализованные финансы

Исследование механик алгоритмического обеспечения, протоколов ликвидности и моделей ценообразования в DeFi-экосистемах

Корпоративные блокчейн-решения

Проектирование архитектуры распределённых реестров для автоматизации B2B-взаимодействия и защищённых расчётов

Инфраструктура и данные

Развитие архитектуры высоконагруженных узлов сетей, разработка децентрализованных оракулов и инструментов анализа данных

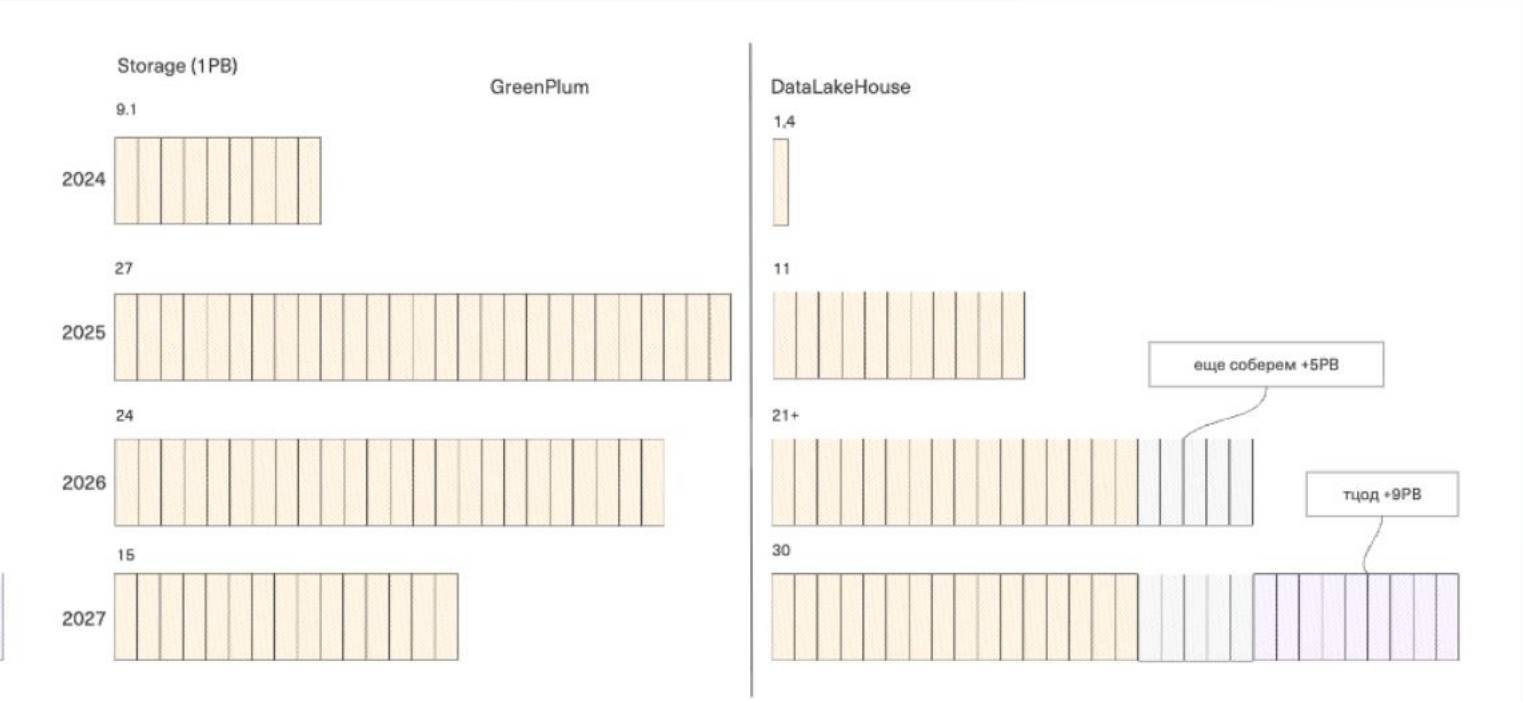
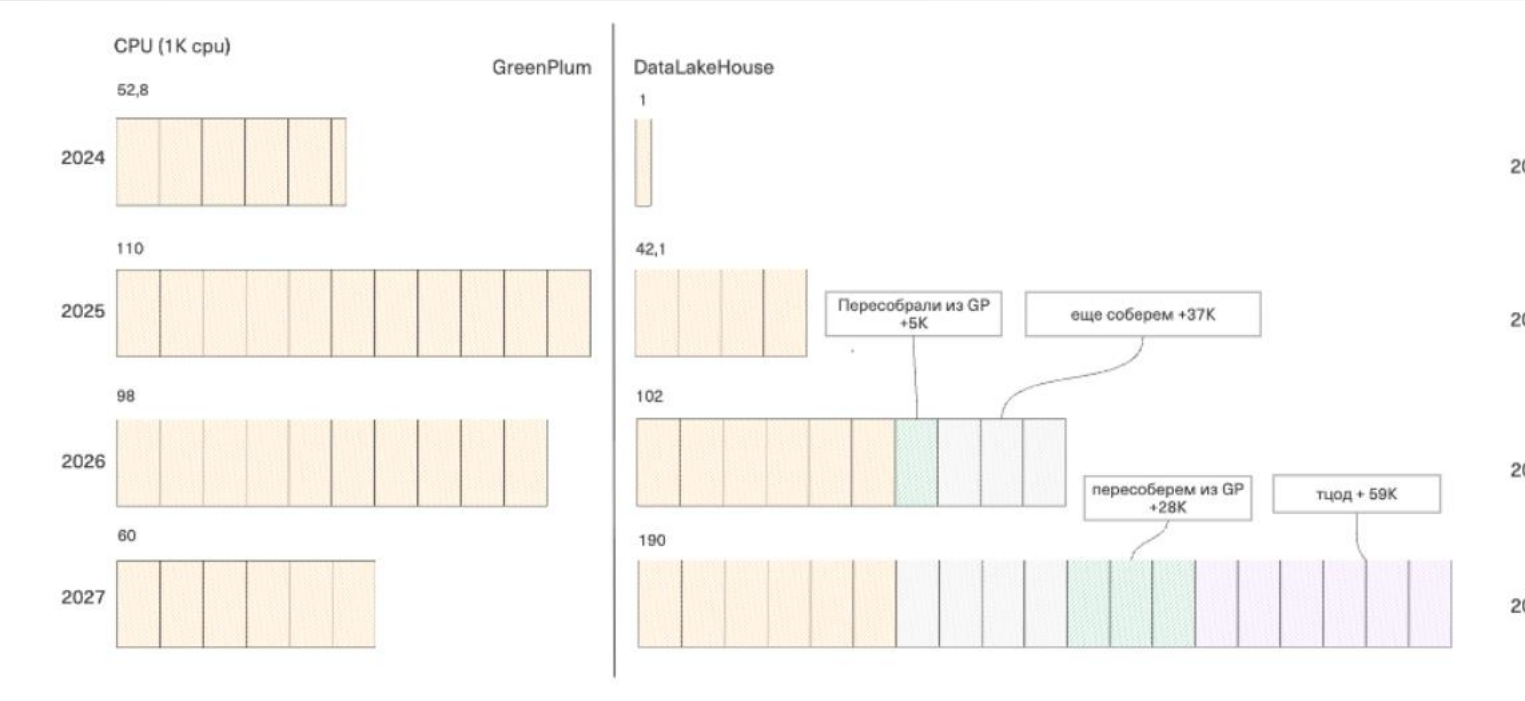
Интеграция Web3 и TradFi

Изучение подходов к бесшовному объединению традиционных финансов с крипто-экосистемами и образовательные инициативы

Спасибо за внимание!

Алексей Теплов
@alekseyteplov

Масштабирование Data Platform



Оптимизация обработки данных и статистические исследования

Медленные и быстрые запросы

70 % времени ожидания и выполнения— приходится на 21%запросов

01	Медленных запросов	21 %
02	Время медленных запросов	70%

* данные за ноябрь и январь 2024



Приближенные вычисления

Объем данных растет, вычислений требуется больше, технологии не успевают за потребностям, требуется альтернативный подход – приближенные вычисления

На уровне данных

уменьшение объема обрабатываемых данных за счет применения статистических методов

На уровне приложения

перфорация циклов, приближенные алгоритмы (HyperLogLog), relaxed synchronization

На аппаратном уровне

специализированные процессоры и системы памяти (например Google TPU)

На уровне схемотехники

приближенный сумматор, умножитель, делитель и т.п.



Виды семплирования

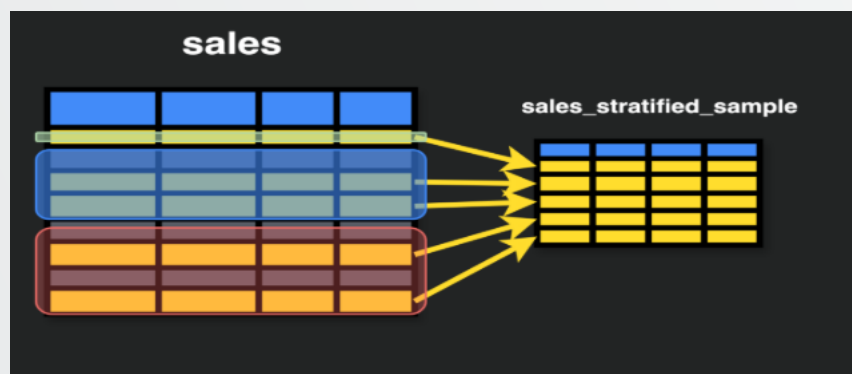
01. Systematic Sampling

элементы выбираются из данных с равными интервалами после случайного начального элемента.



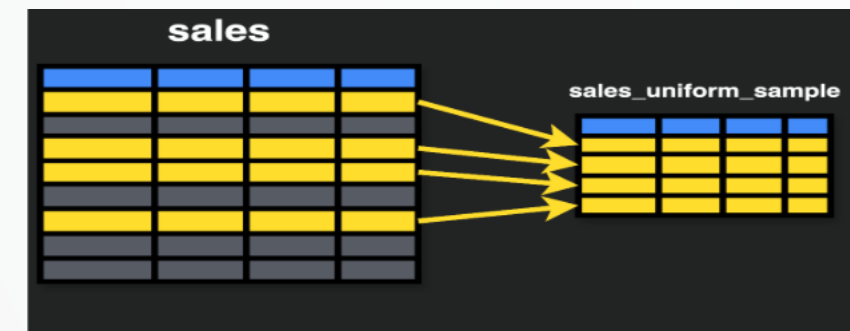
03. Stratified Sampling

данные делятся на группы (страты), из которых выбираются элементы пропорционально их размерам".



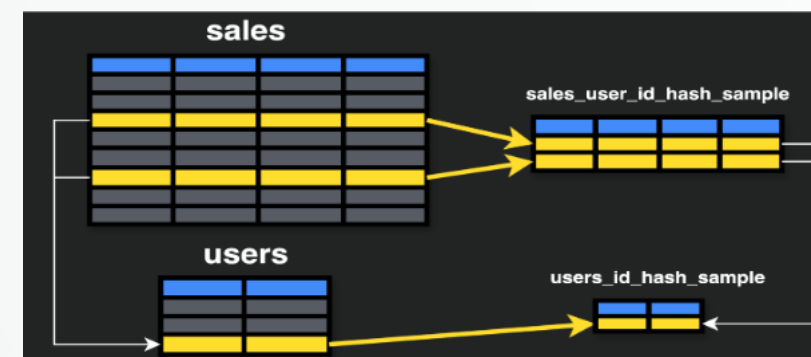
02. Uniform Sampling

элементы данных выбираются равномерно и случайным образом из всей совокупности

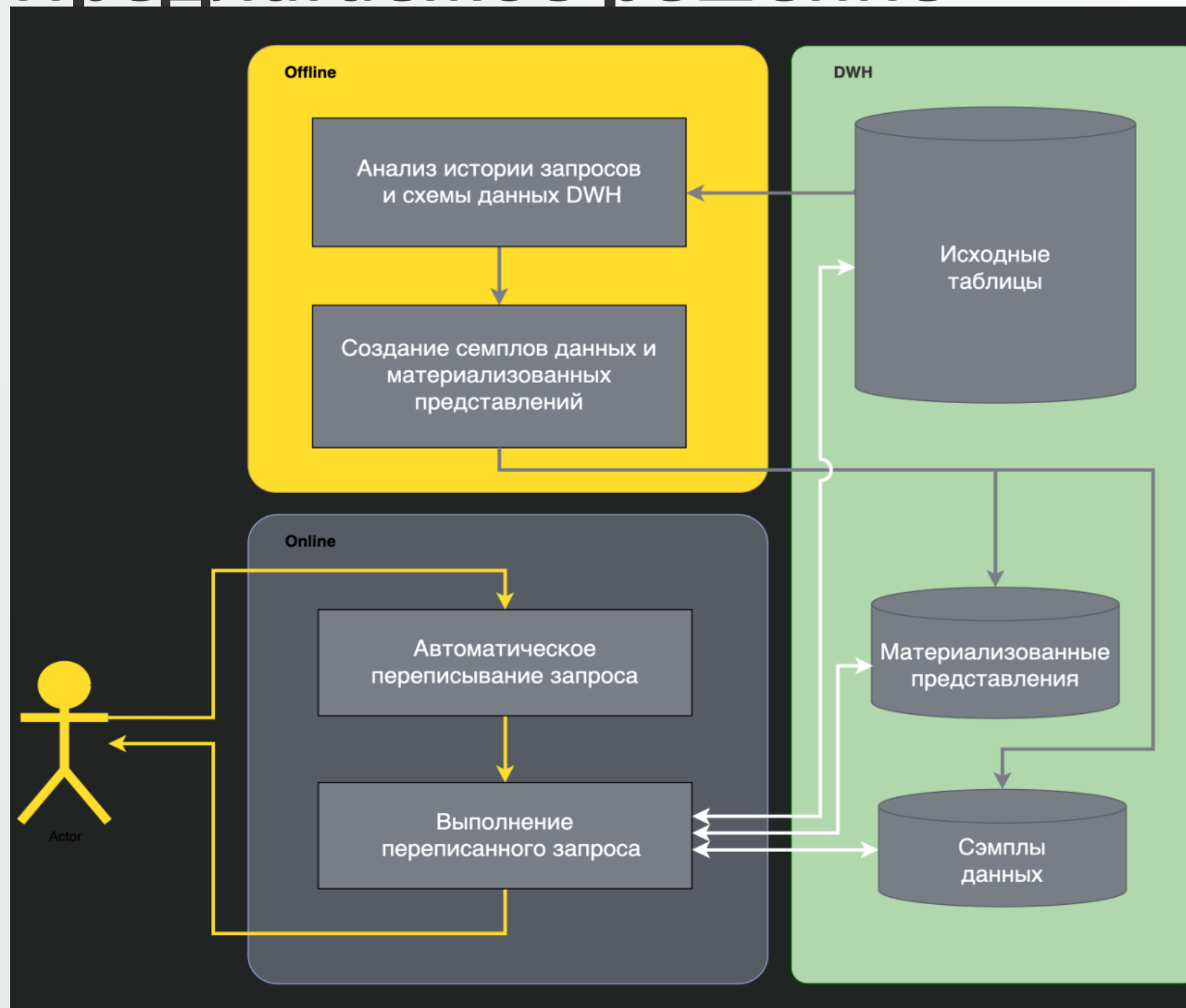


04. Hash Sampling

элементы данных отбираются в выборку на основе значения хэш-функции, примененной к ключевому атрибуту



Предлагаемое решение



Offline

Автоматическая предварительная подготовка материализованных представлений и семплов данных на основе статистики запросов и структуры данных DWH



Online

Обработка запросов пользователей (DBC, ODBC, Helicore)

Запрос автоматически модифицируется для выполнения с использованием DataSketches, материализованных представлений или семплов данных

Bombini : eBPF агент runtime защиты и sandbox AI-нагрузок



- **База на Rust, eBPF и LSM :** сочетает безопасность кода, высокую производительность ядра и гибкость хуков безопасности Linux.
- **Модульная система детекторов:** использует отдельные компоненты для мониторинга процессов, файлов, сети, ядра и io_uring.
- **Фильтрация на уровне ядра:** обрабатывает события внутри eBPF, что минимизирует нагрузку и снижает объем данных для пользователей.
- **Обнаружение продвинутых угроз:** умеет выявлять сложные атаки, включая эксплуатацию уязвимостей через GTFOBins, а также system enumeration.
- **Встроенная песочница (Sandboxing):** позволяет принудительно ограничивать выполнение процессов, доступ к файлам и сетевые соединения.



[github.com/bombinisecurity/
bombini](https://github.com/bombinisecurity/bombini)