Предиктивный анализ паразитной нагрузки на кластерах GreenPlum

Павел Тернюк, Data Sapience

Архитектор

Занимаюсь развитием систем обработки данных на основе open-source продуктов

Марк Лебедев, GlowByte

Архитектор

Строю и развиваю КХД
Занимаюсь развитием open-source GreenPlum

glowbyte

ГлоуБайт — №1 по услугам Business Intelligence и Big Data в России по версии CNews

СБЕР
Alfa-Bank
ГАЗПРОМБАНК
ВТБ
тинькофф
открытие Банк
росбанк
РоссельхозБанк
МОСКОВСКИЙ КРЕДИТНЫЙ БАНК

2300+
специалистов

45+
технологических партнеров

data sapience

Дата Сапиенс — российский разработчик собственных IT-решений, резидент Сколково

4 продуктовых направления:

CM Ocean, Talys.Ocean,
Kolmogorov.ai,
Data Ocean Platform

Разработка

Все продукты являются результатом полной собственной разработки, основанной на глубокой экспертизе команды и многолетнем опыте решения задач наших клиентов

Технологии

Open Source
Frontend – React, Ant Design, JavaScript

Backend – Airflow, Flink, Camunda, Python, Java, PostgreSQL

Развертывание

Платформа Cloud-ready – развертывание в изолированной Kubernetes/Openshift инфраструктуре и облаках

Зачем мы тут сегодня?

* тут могла быть агенда, но только два вопроса нас волнует *

А как бы нам **найти плохие запросы** на кластере GreenPlum еще до того как они станут плохими?

А что вообще можно считать **плохими запросами** для GreenPlum?

Немного про GreenPlum

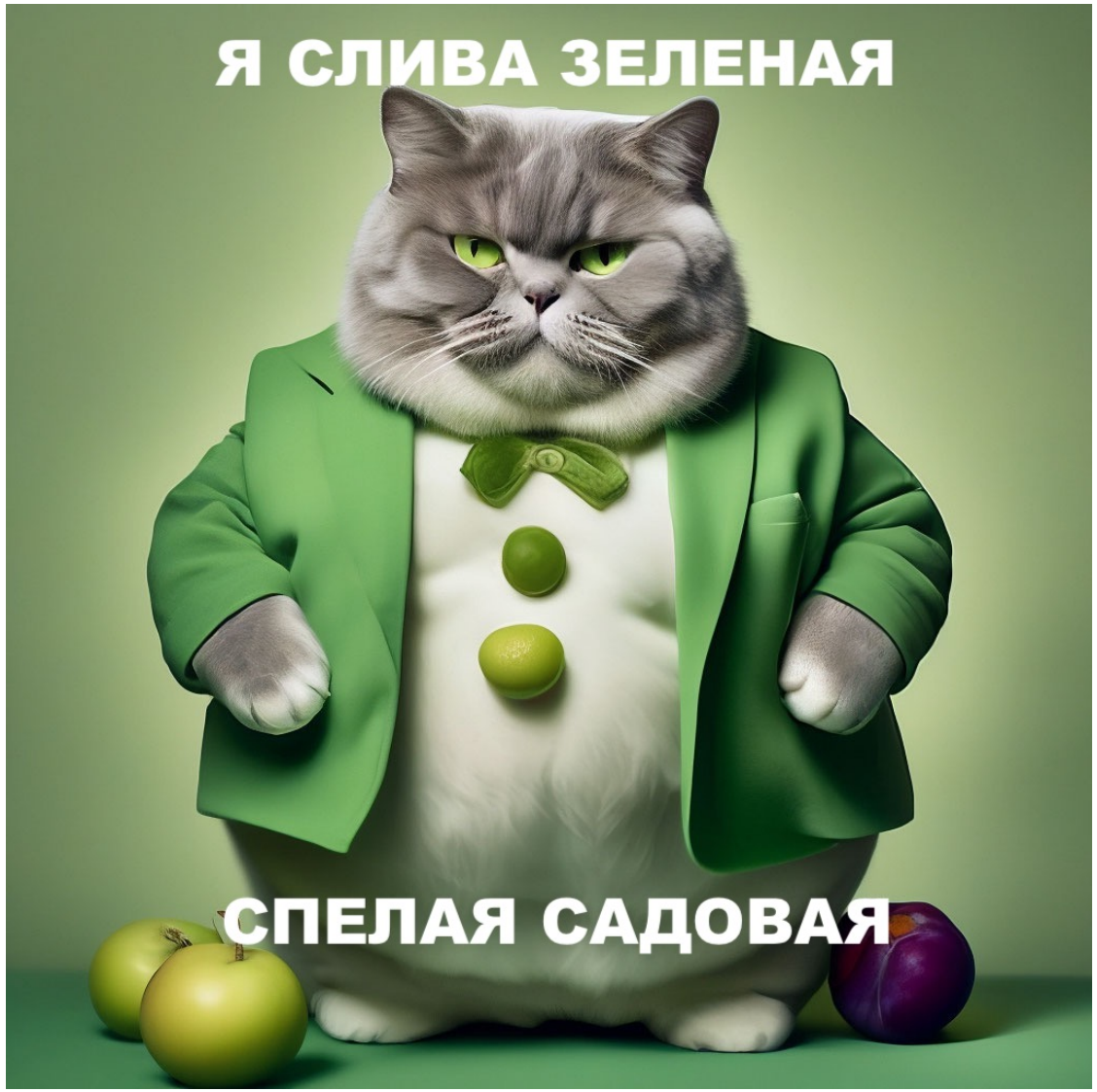
Open-source MPP на базе PostgreSQL 9.4

Набирает популярность в последнее время

Зачастую используется как **ядро КХД**

Горизонтально **масштабируется**

Полноценная поддержка **SQL и ACID**



Особенности GreenPlum

Работает со скоростью самого медленного сегмента

Сильная зависимость от производительности сети и дисков, а это неразделяемые ресурсы

Нуждается в постоянном обслуживании:
- Очистка idle сессий
- Сбор статистики
- Vacuum
- Поиск перекосов
- Отслеживание партиций
- Мониторинг блокировок

Проблемы при работе с GreenPlum

ПРОСТОИ РЕСУРСОВ
Перекосы в нагрузке между сегментами кластера, например по cpu

ОБРАЗОВАНИЕ ОЧЕРЕДЕЙ ЗАПРОСОВ
Исчерпание ресурсных пулов неоптимальными запросами

ПОЛНАЯ НЕРАБОТОСПОСОБНОСТЬ КЛАСТЕРА
Достаточно одного плохого запроса

ОШИБКИ ОПТИМИЗАТОРА
Неактуальность статистики приводит к использованию не оптимальных операций. Самое неприятное - broadcast join

ПАРАЗИТНАЯ НАГРУЗКА НА НЕРАЗДЕЛИМЫЕ РЕСУРСЫ: СЕТЬ, ДИСКИ
Большая редистрибьюция данных или запись временных файлов на диск

Пример паразитной нагрузки. Неверный выбор типа джойна


```
select count(*) from account a2 ;
```

count(*) from account a2

count
10,000,000

```
select count(*) from bills b2 ;
```

count(*) from bills b2

count
10,000,000

	QUERY PLAN
1	Result (cost=0.00..0.00 rows=0 width=0)
2	-> Result (cost=0.00..862.01 rows=1 width=41)
3	-> Redistribute Motion 8:8 (slice2; segments: 8) (cost=0.00..862.00 rows=1 width=41)
4	-> Hash Join (cost=0.00..862.00 rows=1 width=41)
5	Hash Cond: (bills.account_id = account.account_id)
6	-> Seq Scan on bills (cost=0.00..431.00 rows=1 width=41)
7	-> Hash (cost=431.00..431.00 rows=1 width=4)
8	-> Broadcast Motion 8:8 (slice1; segments: 8) (cost=0.00..431.00 rows=1 width=4)
9	-> Seq Scan on account (cost=0.00..431.00 rows=1 width=4)
10	Optimizer: Pivotal Optimizer (GPORCA)

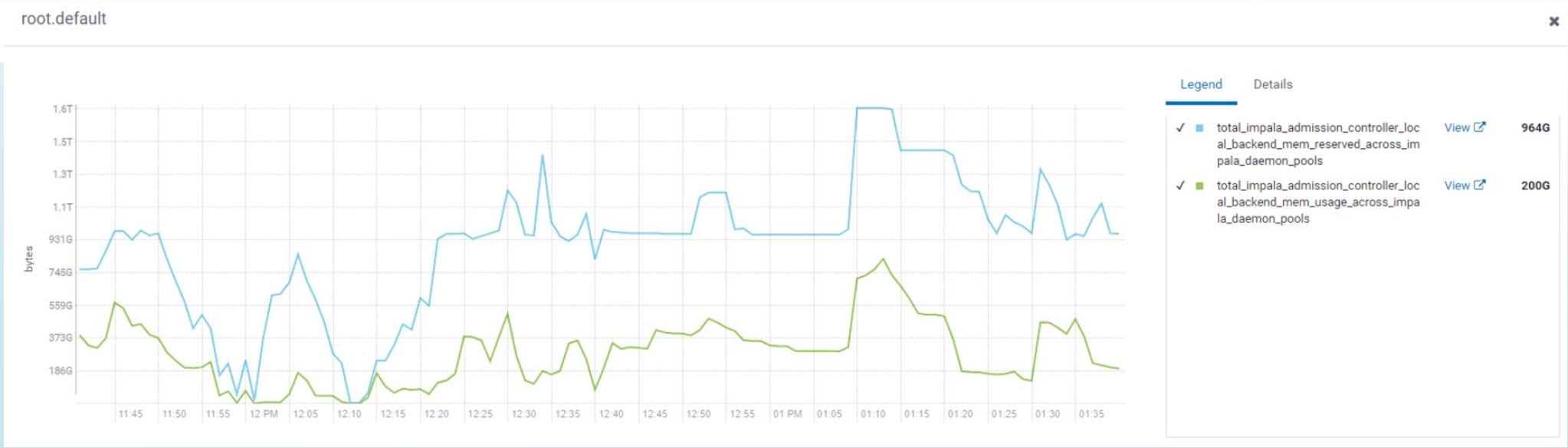
Влияние на бизнес-процессы

Отсутствие хаузкипинга приводит к избыточной инфраструктуре и завышению затрат на обслуживание

Если вообще не вкладываться в обслуживание появляются риски не уложиться в регламент, невыполнение SLA

ИСТОРИЯ ПОХОЖЕЙ ПРОБЛЕМЫ

История решения похожей проблемы на примере Impala



История решения похожей проблемы на примере Impala

Шаблон - периодически повторяющийся запрос (шаг регламентного процесса загрузки/обработки данных)

Запрос - конкретный SQL выполненный движком Impala

шаблоны DIFF > 20Gb	запросы DIFF > 20Gb	шаблоны 20Gb >= DIFF > 15Gb	запросы 20Gb >= DIFF > 15Gb	шаблоны 15Gb >= DIFF > 10Gb	запросы 15Gb >= DIFF > 10Gb
3	42	9	203	51	745
45	708	112	1595	202	2982

В таблице представлено количество запросов с большим переиспользованием памяти, выполненных за неделю.

История решения похожей проблемы на примере Impala

Model	ETL			
	MAE, mb	MSE, mb	Predict+MAE > Peak, %	Predict+MAE > Est, %
LinearRegression	1187.05	2216.51	88.65	52.06
ExtraTreeRegressor	175.8	796.18	92.39	28.5
RandomForestRegressor	211.66	693.62	91.03	29.92
SVR	1186.34	2639.21	78.9	40.33
MLPRegressor	1,753.44	7,887.66	78.86	41.86
Model	**Replication**			
LinearRegression	452.11	1259.73	90.87	24.7
ExtraTreeRegressor	167.87	841.22	91.32	14.17
RandomForestRegressor	112.06	508.44	92.34	14.21
SVR	471.45	1373.41	78.8	14.7
MLPRegressor	671.52	3203.41	80.8	18.39

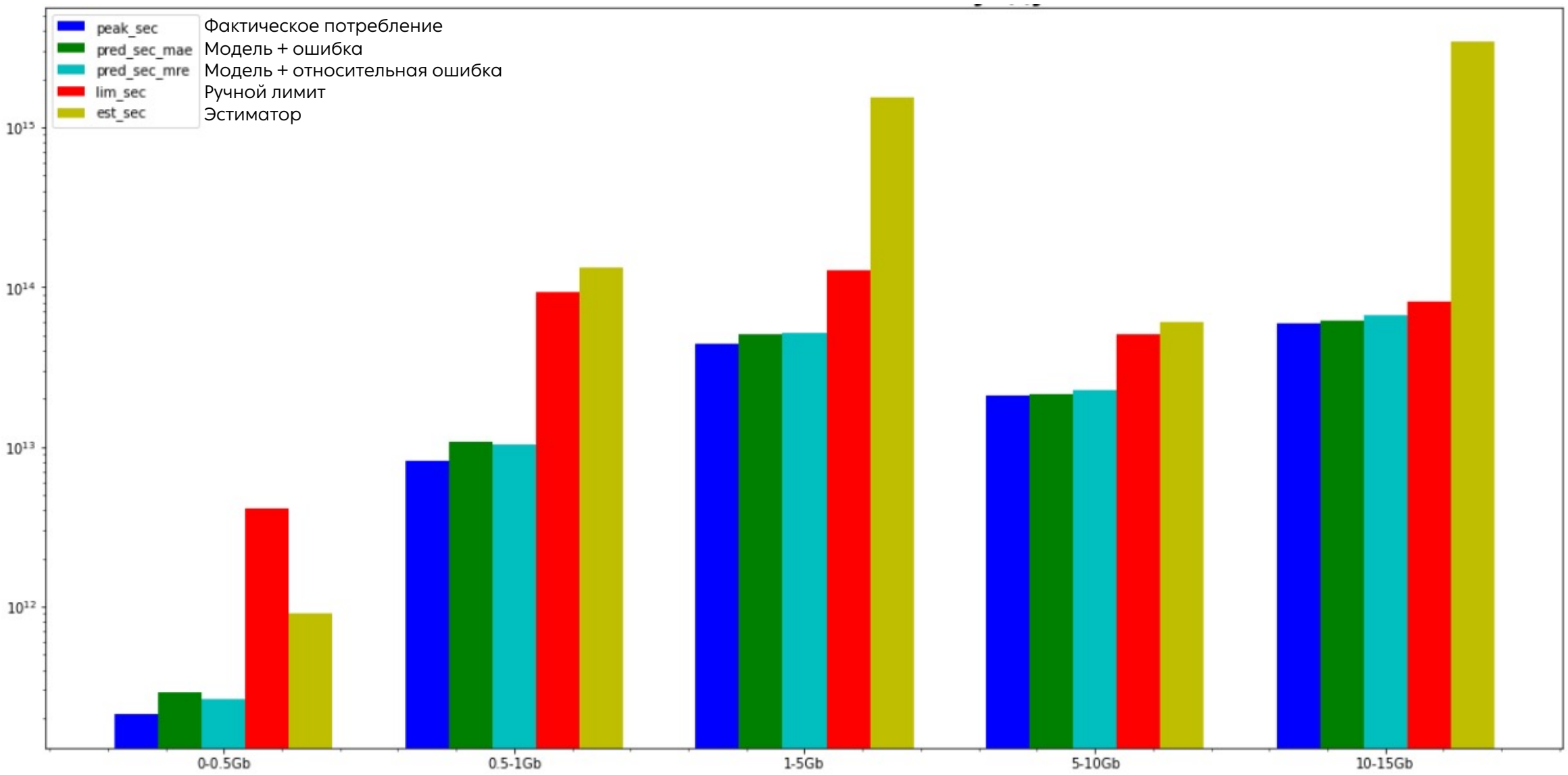
Ретроспективные данные

	ETL			
	MAE, mb	MSE, mb	Predict+MAE > Peak, %	Predict+MAE > Est, %
Без модификаций	175.8	796.18	92.39	28.5
Фильтрация	210.07	886.81	91.52	17.50
Модификация ошибки	175.8	796.18	86.15	9.21
	Replication			
Без модификаций	167.87	841.22	91.34	14.17
Фильтрация	176.20	809.16	91.22	7.42
Модификация ошибки	167.87	841.22	85.45	2.03

Пилот на реальном хранилище

- Модель показывает себя на "больший запросах"
- 8 из 10 предсказаний лучше эстиматора
- Менее 3% запросов завершается с ошибкой по памяти

История решения похожей проблемы на примере Impala



Возвращаемся к GreenPlum

А как перенести полученный опыт на GreenPlum?

Для начала нужны метрики

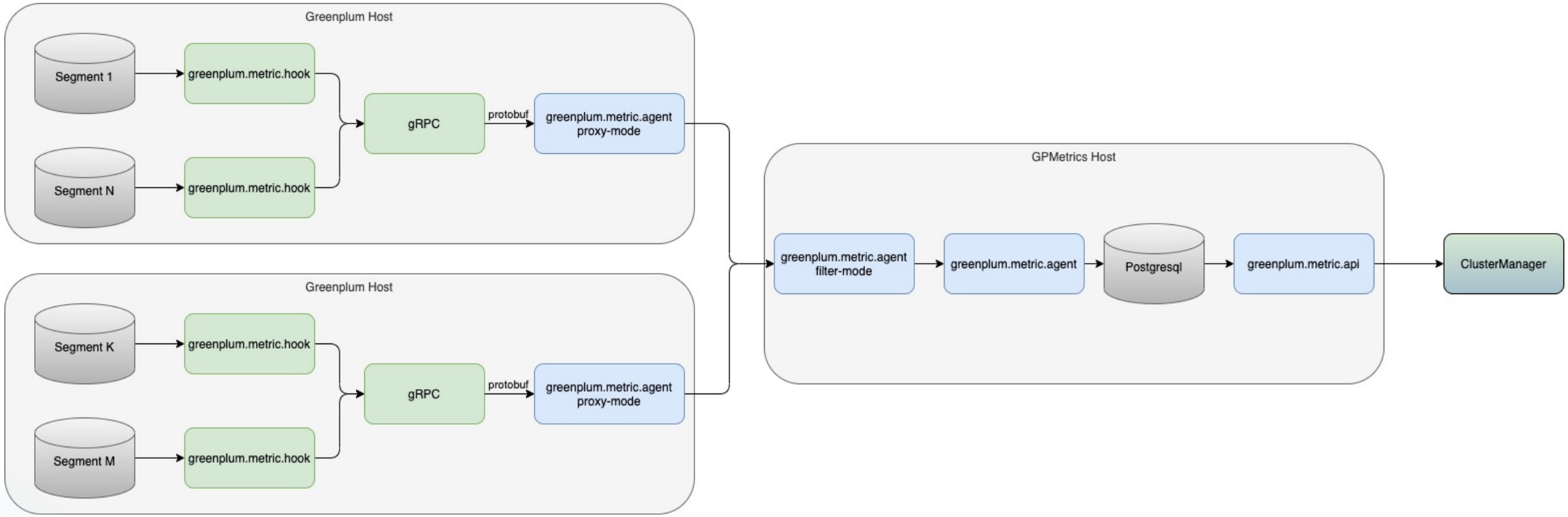
Инструменты мониторинга OS GreenPlum

А их нет...

Инструменты мониторинга OS GreenPlum

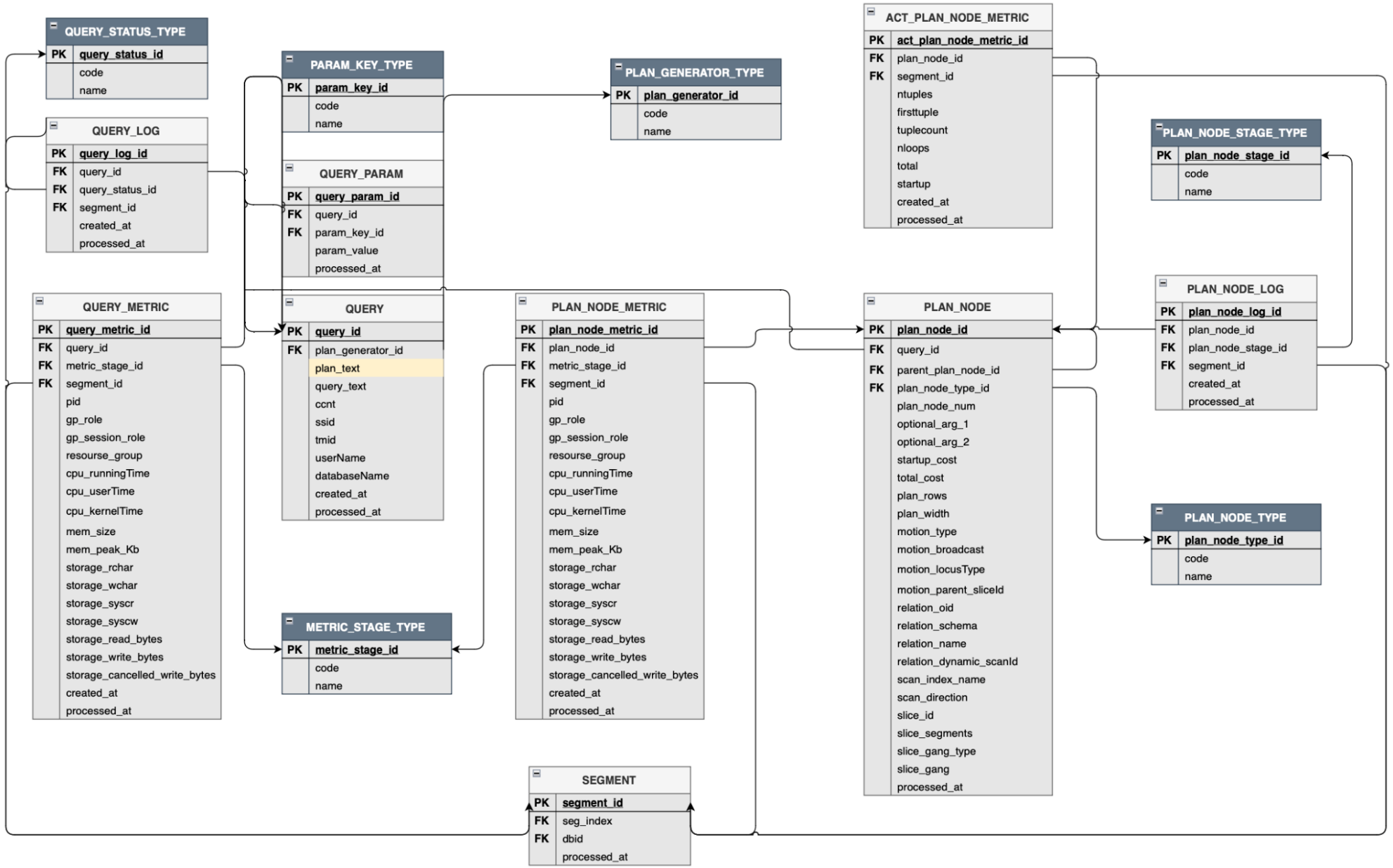
... но мы попытались это исправить.
Выложили в открытый доступ библиотеку хуков

Как мы собираем метрики



GreenPlum,
PostgreSQL, Hooks

Модель метрик



А как перенести полученный опыт на GreenPlum?

Метрики

Критерии определения паразитных запросов

Критерии определения паразитных запросов. На старте запроса

Показатель нагрузки на Master

Критерии определения паразитных запросов. На старте запроса

Показатель нагрузки на Master

Фактор сбора статистики

Критерии определения паразитных запросов. На старте запроса

Показатель нагрузки на Master

Фактор сбора статистики

Фактор наличия nested loop join

Критерии определения паразитных запросов. На старте запроса

Показатель нагрузки на Master

Фактор сбора статистики

Фактор наличия nested loop join

Фактор сложности запроса

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Показатель отклонения планового и фактического количества строк

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Показатель отклонения планового и фактического количества строк

Показатель перекоса данных

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Показатель отклонения планового и фактического количества строк

Показатель перекоса данных

Показатель простоя CPU

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Показатель отклонения планового и фактического количества строк

Показатель перекоса данных

Показатель простоя CPU

Фактор записи временных файлов (spill)

Критерии определения паразитных запросов. В рантайме

Объем перераспределения данных

Показатель отклонения планового и фактического количества строк

Показатель перекоса данных

Показатель простоя CPU

Фактор записи временных файлов (spill)

Фактор сложности запроса

Примеры. Фактор записи временных файлов


```
SELECT pn.query_id, q.query_text, date(q.created_at) created_at,
            sum(pnm.spill_file_count) AS spill_file_count,
            sum(pnm.spill_total_size) AS spill_total_size
           FROM greenplum_metric.plan_node pn
           JOIN greenplum_metric.plan_node_metric pnm ON pn.plan_node_id = pnm.plan_node_id
           join greenplum_metric.query q on pn.query_id = q.query_id
           where
q.created_at >= '2023-04-19' and q.created_at < '2023-04-20'
and pnm.spill_file_count > 0 or pnm.spill_total_size > 0
          GROUP BY 1,2,3
          order by query_id;
```

	query_id	query_text	created_at	spill_file_count	spill_total_size
1	171	/* {"app": "dbt", "dbt_versic	2023-03-23	57	285 474 816
2	197	/* {"app": "dbt", "dbt_versic	2023-03-23	56	1 278 705 664
3	216	/* {"app": "dbt", "dbt_versic	2023-03-23	201	6 664 847 360
4	374	/* {"app": "dbt", "dbt_versic	2023-03-23	320	1 720 320 000
5	389	/* {"app": "dbt", "dbt_versic	2023-03-23	160	1 501 626 368
6	422	/* {"app": "dbt", "dbt_versic	2023-03-23	644	3 271 622 712
7	447	/* {"app": "dbt", "dbt_versic	2023-03-23	420	6 881 280
8	463	/* {"app": "dbt", "dbt_versic	2023-03-23	41 604 365	731 487 435 524
9	816	/* {"app": "dbt", "dbt_versic	2023-04-05	57	295 534 592
10	842	/* {"app": "dbt", "dbt_versic	2023-04-05	56	1 280 245 760
11	861	/* {"app": "dbt", "dbt_versic	2023-04-05	205	7 346 552 832
12	1 019	/* {"app": "dbt", "dbt_versic	2023-04-05	320	1 720 942 592
13	1 034	/* {"app": "dbt", "dbt_versic	2023-04-05	128	1 465 122 816
14	1 067	/* {"app": "dbt", "dbt_versic	2023-04-05	616	3 184 248 644

Примеры. Показатель перекоса данных


```
SELECT a.query_id, a.query_text, a.created_at, max(a.skew_rows) * 100::double precision AS skew_volume from
(SELECT pn.query_id, q.query_text, pn.plan_node_id, apnm.ntuples, apnm.nloops , date(q.created_at) as created_at,
(max(apnm.ntuples/apnm.nloops) OVER (PARTITION BY pn.query_id, pn.plan_node_id) - min(
                        CASE
                            WHEN apnm.ntuples = 0 THEN NULL::bigint
                            ELSE apnm.ntuples/apnm.nloops::double precision
                        END) OVER (PARTITION BY pn.query_id, pn.plan_node_id))::double precision / max(
                        CASE
                            WHEN apnm.ntuples = 0 THEN 1::bigint
                            ELSE apnm.ntuples/apnm.nloops::double precision
                        END) OVER (PARTITION BY pn.query_id, pn.plan_node_id)::double precision AS skew_rows
                       from greenplum_metric.plan_node pn
                       join greenplum_metric.query q on pn.query_id = q.query_id
                       join greenplum_metric.act_plan_node_metric apnm on pn.plan_node_id = apnm.plan_node_id
                       join greenplum_metric.plan_node_type pnt on pn.plan_node_type_id = pnt.plan_node_type_id
                       where pnt.name != 'Result'
                       GROUP BY 1,2,3,4,5,6) a where a.created_at >= '2023-03-23' and a.created_at < '2023-03-24'
 GROUP BY 1,2,3
order by skew_volume
```

query_id	query_text	created_at	skew_volume
463	/* {"app": "dbt", "dbt_version": "1.2.0", "profile_name": "gp_d	2023-03-23	99,9777692214
422	/* {"app": "dbt", "dbt_version": "1.2.0", "profile_name": "gp_d	2023-03-23	97,4793256997
447	/* {"app": "dbt", "dbt_version": "1.2.0", "profile_name": "gp_d	2023-03-23	97,2633693196
77	/* {"app": "dbt", "dbt_version": "1.2.0", "profile_name": "gp_d	2023-03-23	93,9170182841

А как перенести полученный опыт на GreenPlum?

Метрики

Критерии

Модель

Классификация запросов и получение пороговых значений для выбранных метрик и критериев

Данные для обучения

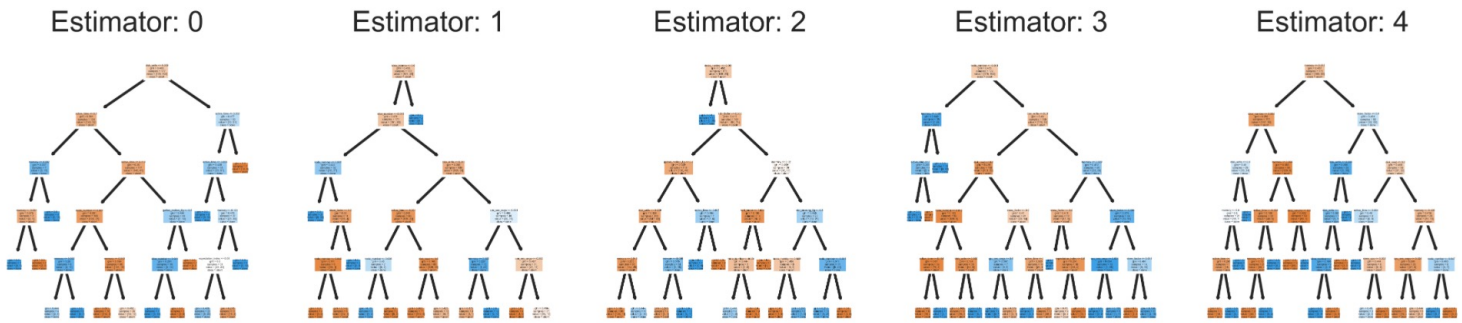
- Данные с реальных кластеров. Деление запросов по реакции администраторов/разработчиков
- Генерация синтетических данных и запросов с типичными проблемами

Процесс обучения модели и выбора ее типа

- Нормирование неограниченных показателей на ресурсы кластера
- Подход с классификацией
- Обучение и периодическая актуализация ее весов производиться на истории конкретного кластера (и синтетики на нем же)

Классификация запросов и получение пороговых значений для выбранных метрик и критериев


```
columns_for_model = ['stat_missing_flg',
                     'gather_motion_flg',
                     'nested_loop_flg',
                     'slice_volume',
                     'slice_number',
                     'node_number',
                     'distirbuted_index',
                     'expectation_index',
                     'skew_factor',
                     'disk_read',
                     'disk_write',
                     'cpu_sec_master',
                     'cpu_sec_segs',
                     'skew_cpu',
                     'broadcast_index',
                     'spill_factor',
                     'memory',
                     'skew_rows',
                     'cost',
                     'active_time',
                    ]
columns_for_target = ['status']
```



stat_missing_flg: 0.00
gather_motion_flg: 0.01
nested_loop_flg: 0.01
slice_volume: 0.01
slice_number: 0.06
node_number: 0.22
distirbuted_index: 0.00
expectation_index: 0.02
skew_factor: 0.11
disk_read: 0.00
disk_write: 0.04

cpu_sec_master: 0.03
cpu_sec_segs: 0.00
skew_cpu: 0.06
broadcast_index: 0.03
spill_factor: 0.02
memory: 0.31
skew_rows: 0.00
cost: 0.02
active_time: 0.05

А как перенести полученный опыт на GreenPlum?

Метрики

Критерии

Модель

Реакция

Варианты реакции на паразитную нагрузку

Прерывание запроса

Варианты реакции на паразитную нагрузку

Прерывание запроса

Перемещение в другую ресурсную группу

Варианты реакции на паразитную нагрузку

Прерывание запроса

Перемещение в другую ресурсную группу

Рекомендации по обслуживанию БД

Варианты реакции на паразитную нагрузку

Прерывание запроса

Перемещение в другую ресурсную группу

Рекомендации по обслуживанию БД

Блокировка нарушителей

Варианты реакции на паразитную нагрузку

Прерывание запроса

Перемещение в другую ресурсную группу

Рекомендации по обслуживанию БД

Блокировка нарушителей

Изменение параметров сессии при запуске запроса (*)

Варианты реакции на паразитную нагрузку

Прерывание запроса

Перемещение в другую ресурсную группу

Рекомендации по обслуживанию БД

Блокировка нарушителей

Изменение параметров сессии при запуске запроса (*)

Внесение изменений в план выполнения запроса (*)

Планы на будущее

Завершить MVP 1.0

- Придумать новые метрики
- Научиться управлять параметрами сессии
- Научиться влиять на план запроса

Интеграции с другими инструментами

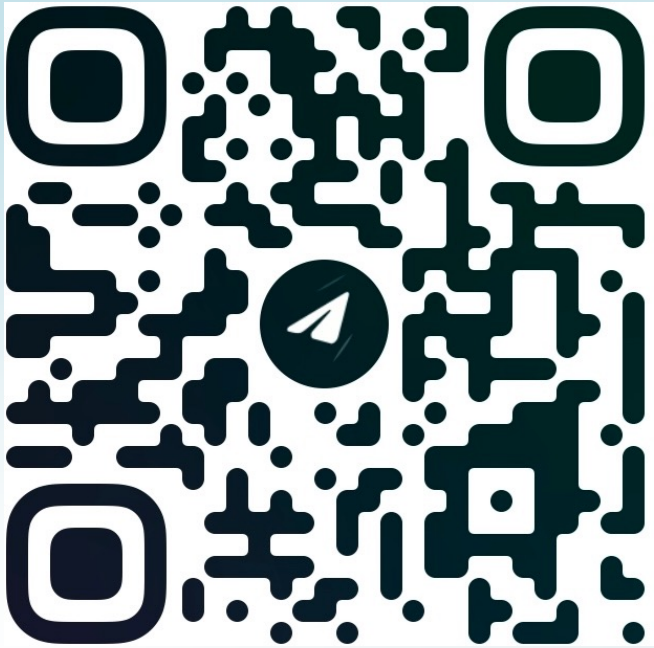
- Научиться управлять механизмом через Cluster Manager
- Научиться рисовать отчеты по паразитности

Финализируем

Не забывайте обслуживать свой GreenPlum

Храните историю запросов

Даже простые ML-модели могут помочь автоматизировать процессы обслуживания



Контакты

mark.lebedev@glowbyteconsulting.com
pavel.ternyuk@glowbyteconsulting.com
contact@datasapience.com