

Greenplum: командный центр вместо pg_stat_statements

Леонид Борчук,
Team Lead Greenplum,
Yandex Cloud

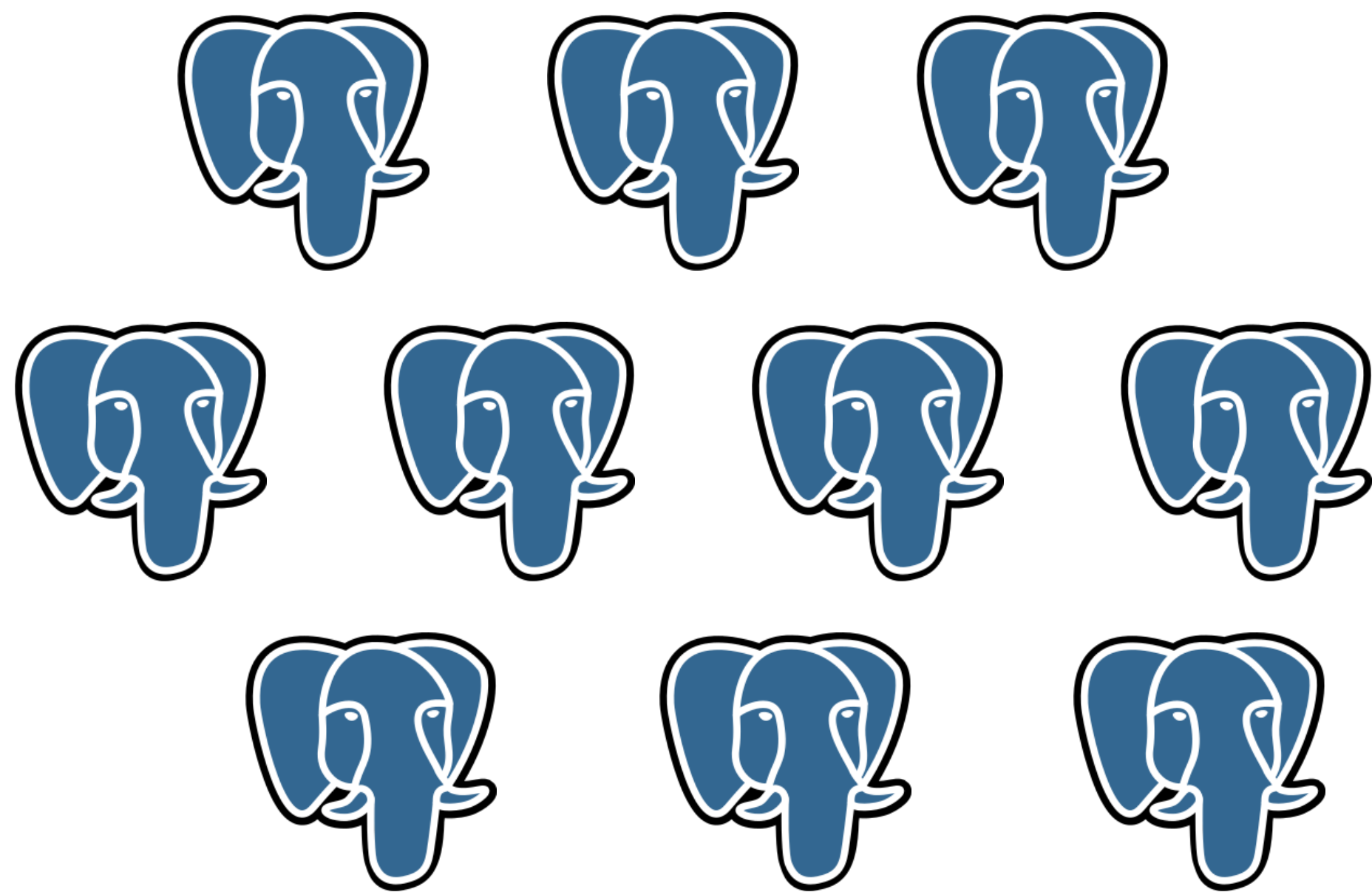
Наша
команда



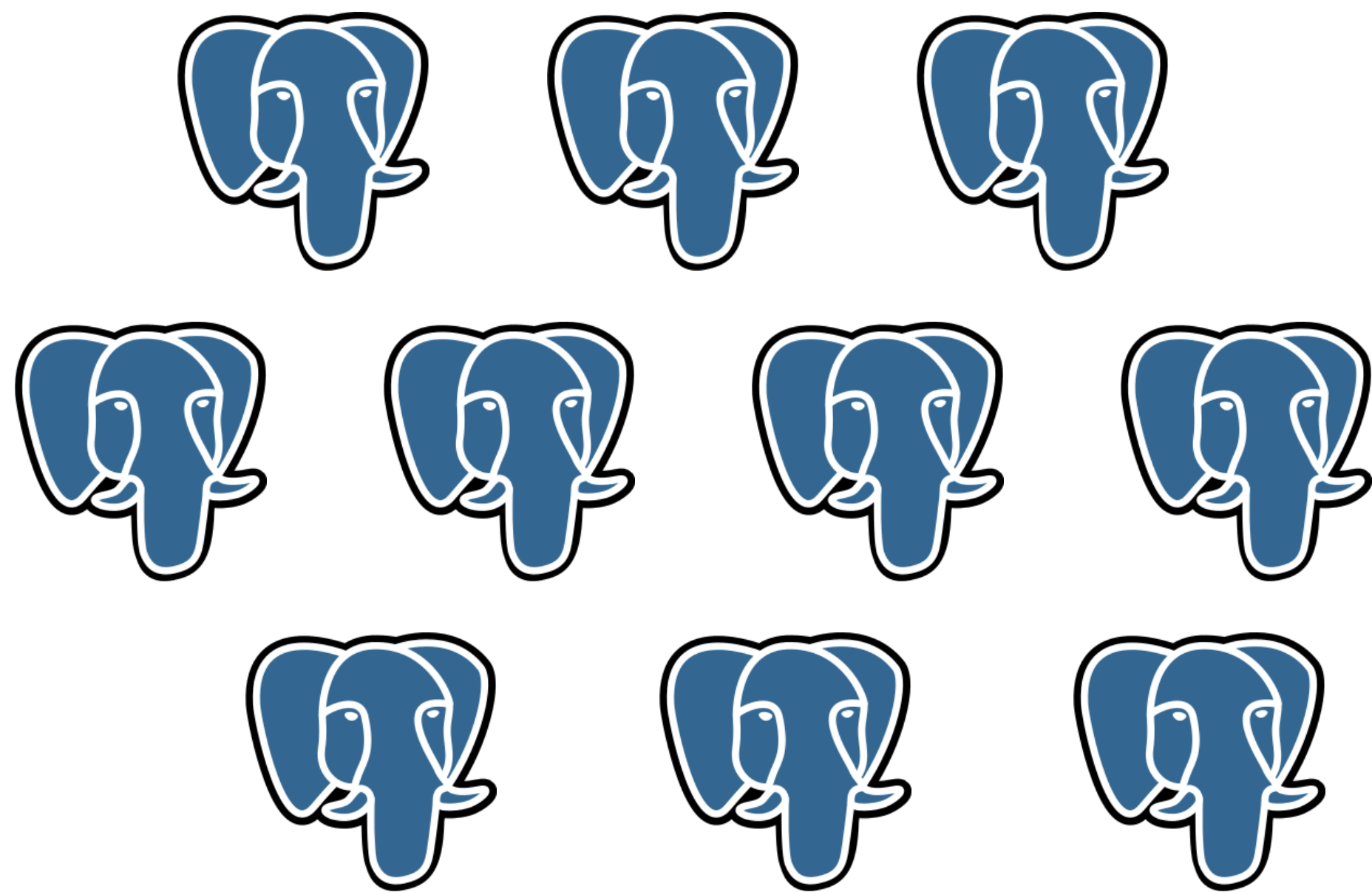
О чём сегодня поговорим

1. Что такое командный центр
2. Extension на замену pg_stat_statements
3. Автобатчинг и дропы данных
4. Полезные возможности и недостатки

1. Что такое командный центр
2. Extension на замену
pg_stat_statements
3. Автобатчинг и дропы данных
4. Полезные возможности и недостатки



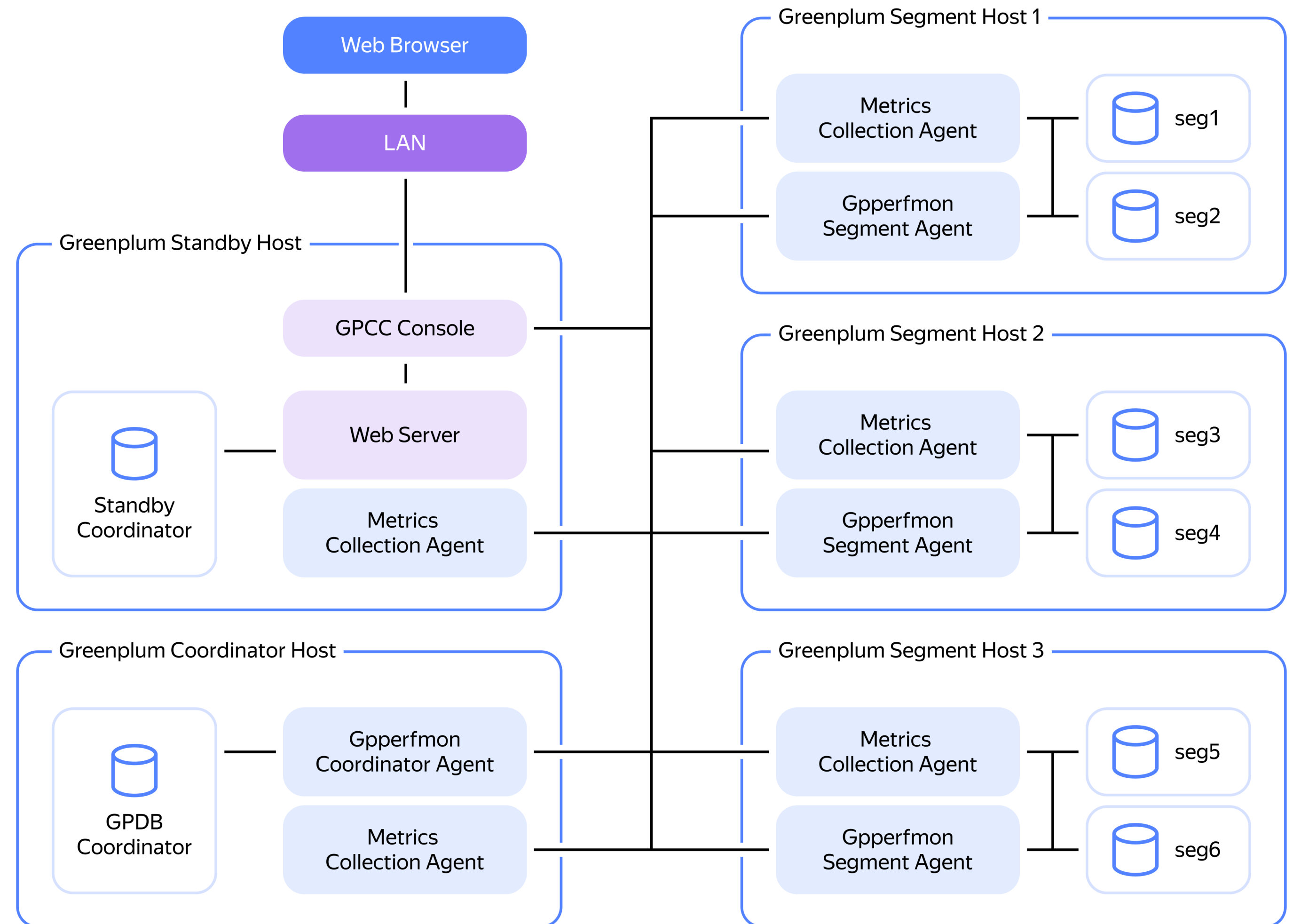
Greenplum —
это же просто
много Postgres'ов!



Почему
не pg_stat_statements

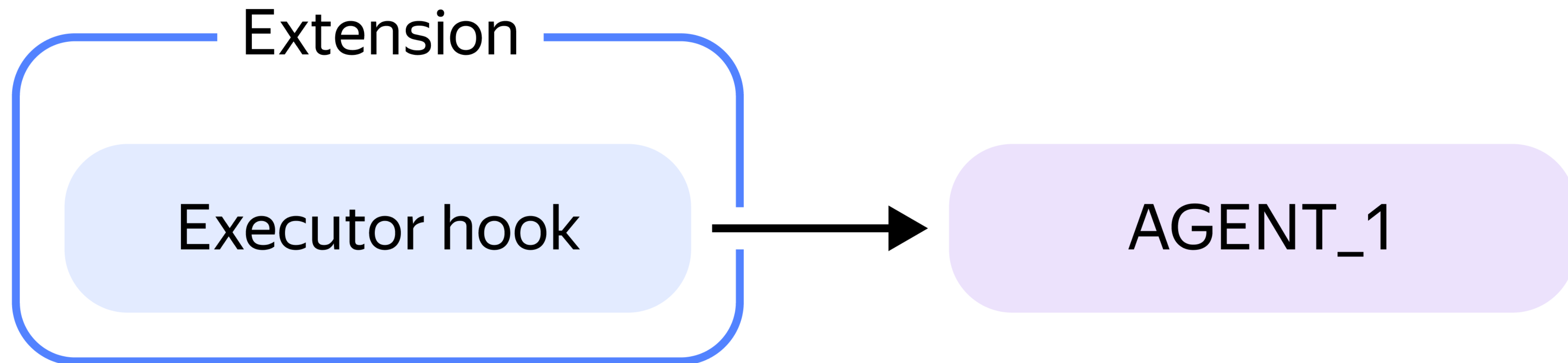
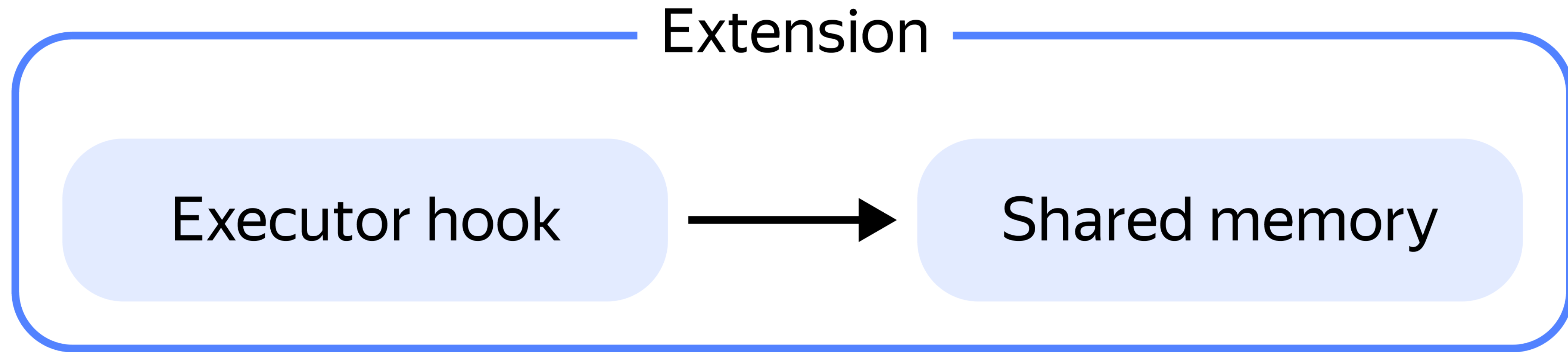
Командный центр

1. Управление кластерами
(как в Oracle EM)
2. Сбор performance-метрик
с кластера
3. Сбор статистики
выполнения запросов
4. Управление
ресурсными группами

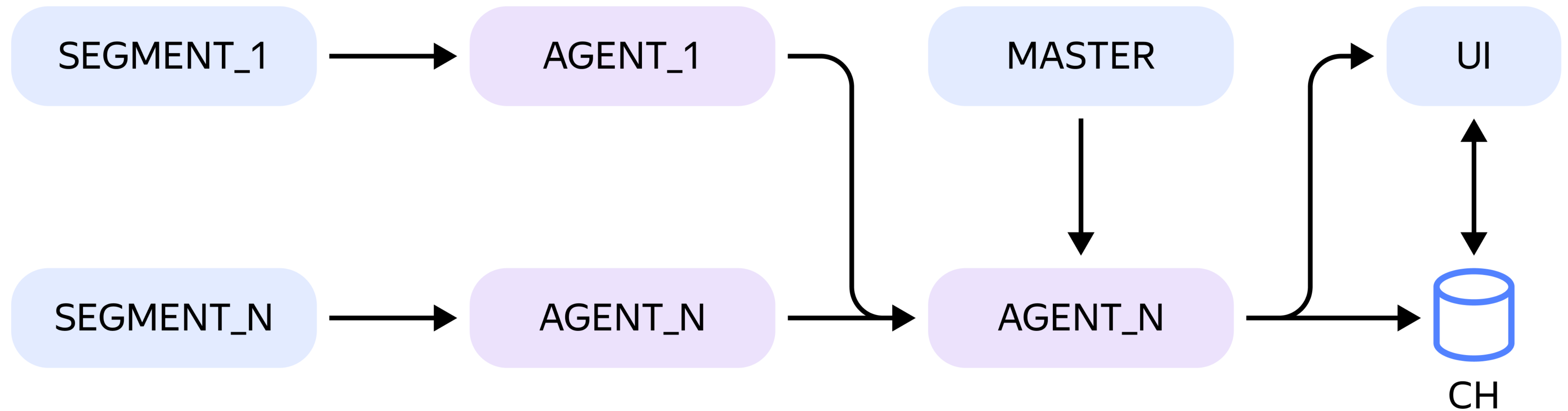


1. Что такое командный центр
2. Extension на замену
`pg_stat_statements`
3. Автобатчинг и дропы данных
4. Полезные возможности и недостатки

Что изменилось



Архитектура



Extension PG → Agent

```
void ExecutorStart_hook(QueryDesc * queryDesc)
```

```
void ExecutorRun_hook(QueryDesc * queryDesc,  
ScanDirection direction, long count)
```

```
void ExecutorFinish_hook(QueryDesc * queryDesc)
```

```
void ExecutorEnd_hook(QueryDesc * queryDesc)
```

```
void query_info_collect_hook  
(QueryMetricsStatus status, void * args)
```

Extension PG → Agent ^{Highload}



Greenplum CC — RPC
clck.ru/39pkkW



GlowByte CC — GRPC
clck.ru/39pmK9

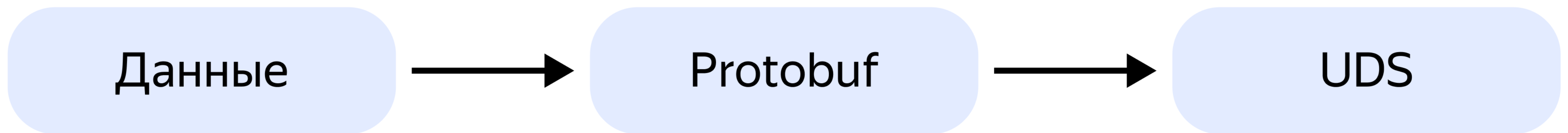
GRPC — наивная реализация

- Postgres — не thread-safe
- GRPC — асинхронная отправка

GRPC — наивная реализация

- GRPC перехватывает сигналы
- Много соединений, и установка/разрыв соединения асинхронны
- Сложно сделать автобатчинг и дроп пакетов

Итоговое решение



```
$ go test -bench=. -benchmem
```

```
cpu: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
```

BenchmarkQueryRandFloat-12	21021	51643	ns/op	9560	B/op	177	allocs/op
BenchmarkQuerySameValue-12	21942	52699	ns/op	9555	B/op	177	allocs/op
BenchmarkNodeRandFloat-12	22423	52413	ns/op	9809	B/op	179	allocs/op

1. Что такое командный центр
2. Extension на замену
`pg_stat_statements`
3. Автобатчинг и дропы данных
4. Полезные возможности и недостатки

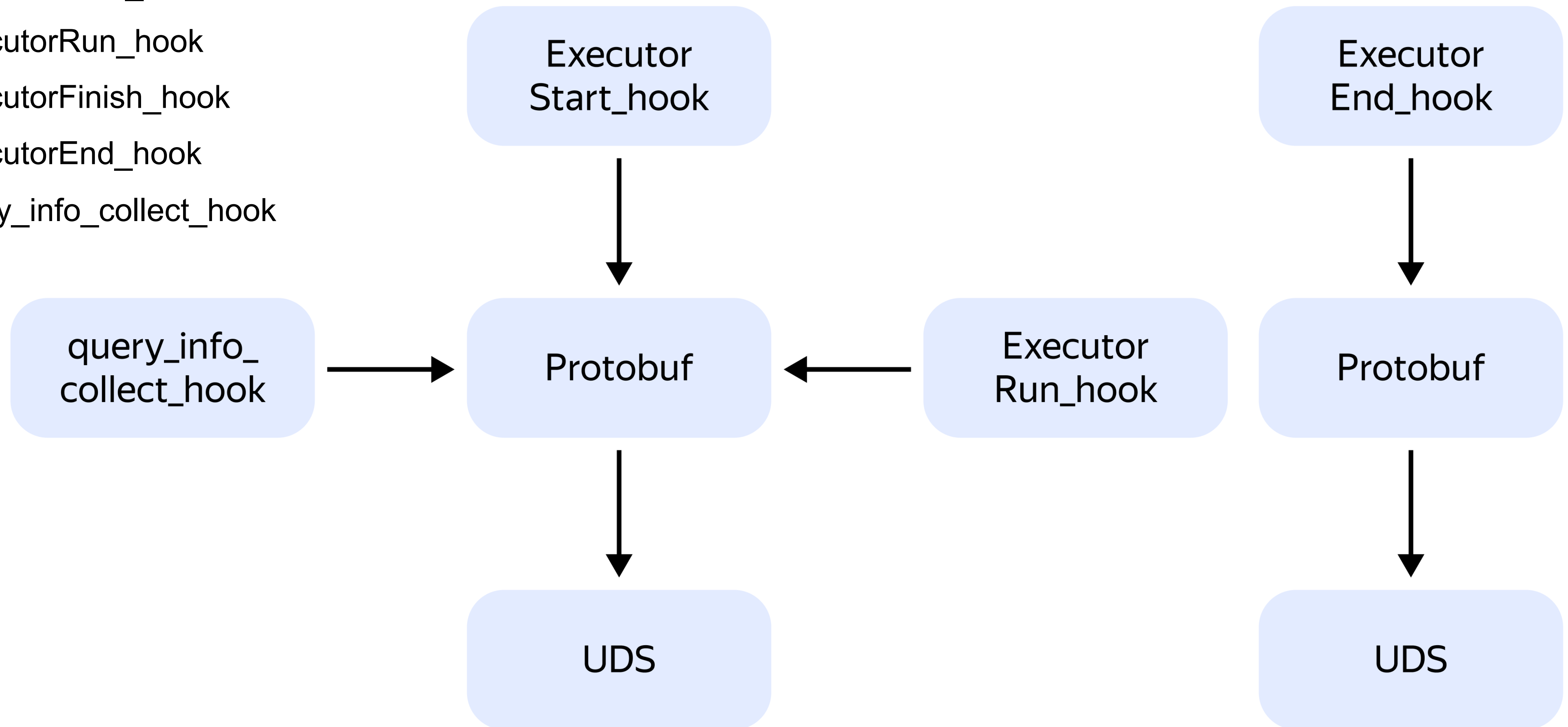
Автобатчинг

- Отправка нескольких сообщений одним запросом
- Нужен на всех этапах!
- **Merge**, не log и не replace



Автобатчинг GP → Агент

- ExecutorStart_hook
- ExecutorRun_hook
- ExecutorFinish_hook
- ExecutorEnd_hook
- query_info_collect_hook



Дроп данных GP → Агент

Агент перегружен или недоступен

`O_NONBLOCK` or `O_NDELAY`. When possible, the file is opened in nonblocking mode. Neither the `open()` nor any subsequent I/O operations on the file descriptor which is returned will cause the calling process to wait

ExecutorEnd → Delay → Drop

Автобатчинг Segment → Master



func MaxMsgSize DEPRECATED [Hide](#)

```
func MaxMsgSize(m int) ServerOption
```

MaxMsgSize returns a ServerOption to set the max message size in bytes the server can receive. If this is not set, gRPC uses the default limit.

Deprecated: use MaxRecvMsgSize instead. Will be supported throughout 1.x.

Дроп данных Segment/Master

[postgres](#) / [contrib](#) / [pg_stat_statements](#) / [pg_stat_statements.c](#)

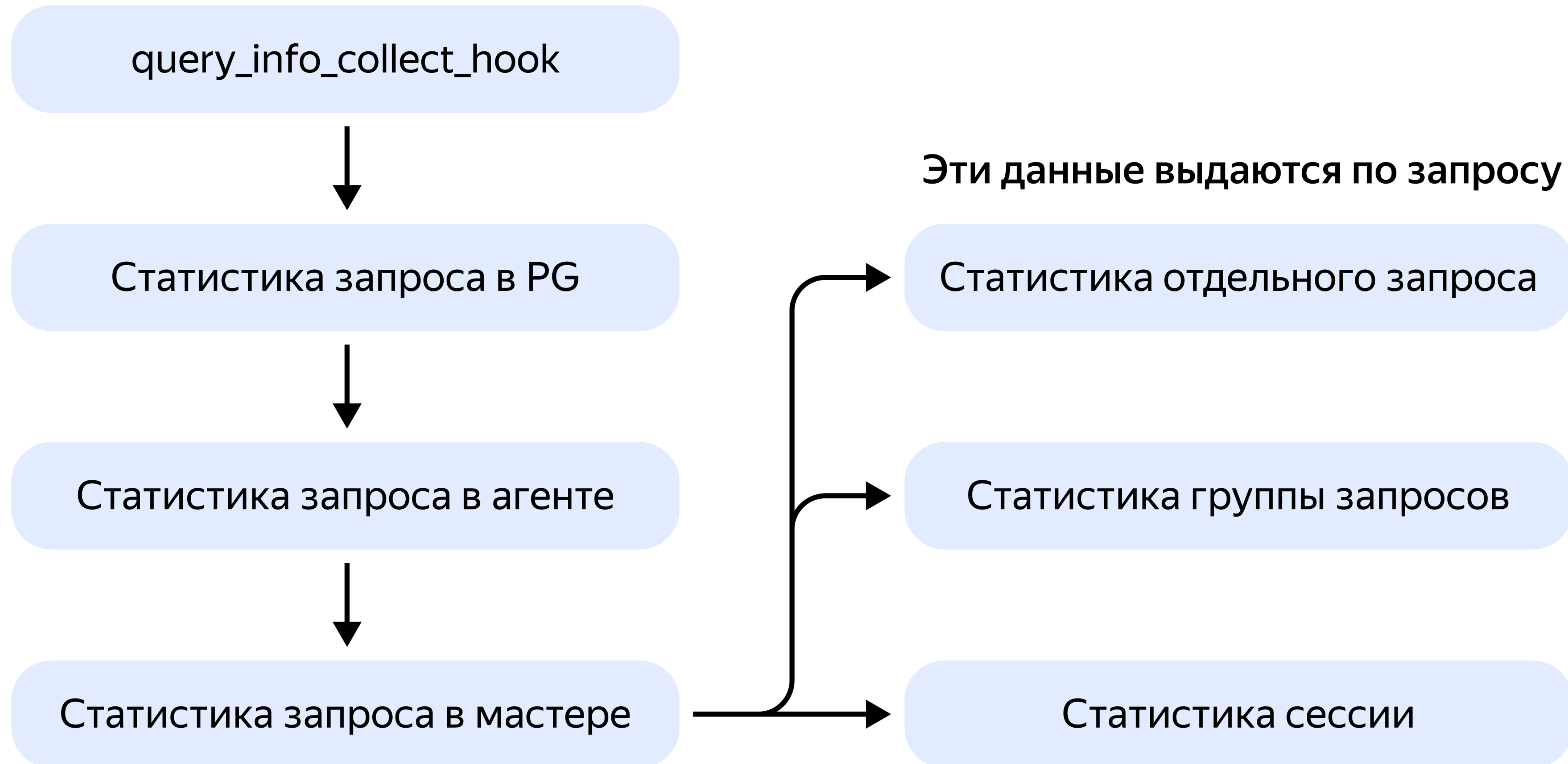
Code

Blame

3012 lines (2646 loc) · 87 KB

```
212      typedef struct pgssGlobalStats
-----
2097      /*
2098      * Deallocate least-used entries.
2099      *
2100      * Caller must hold an exclusive lock on pgss->lock.
2101      */
2102      static void
2103      entry_dealloc(void)
2104      {
```

Батчинг → Агрегация



Дроп данных Master

Данные старых сессий
(периодически опрашиваем
pg_stat_activity)



Данные с сегментов
об удалённых запросах



Данные завершённых
запросов



Частичные данные
от агента



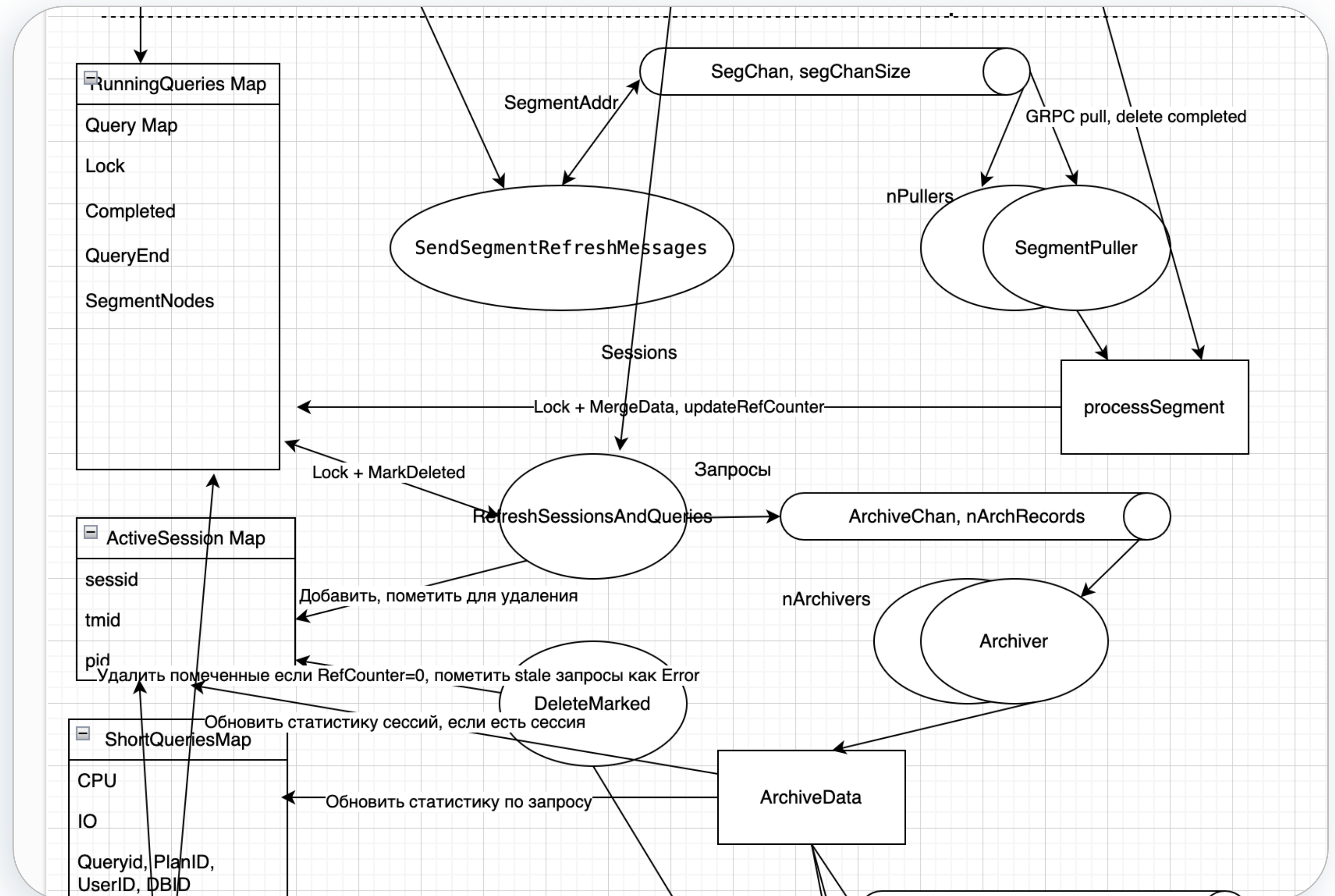
1. Что такое командный центр
2. Extension на замену
`pg_stat_statements`
3. Автобатчинг и дропы данных
4. Полезные возможности и недостатки

Что это даёт

Современный язык программирования

- Data race detector
- Deadlock
- Benchmark tests

Написали приложение вдвоём



Что это даёт

1

Сложная логика
агрегации

2

Объединение
данных из разных
источников

3

GRPC-интерфейс
для общения
с внешними
компонентами

Что это даёт

1

Отгрузка данных
в хранилище метрик

2

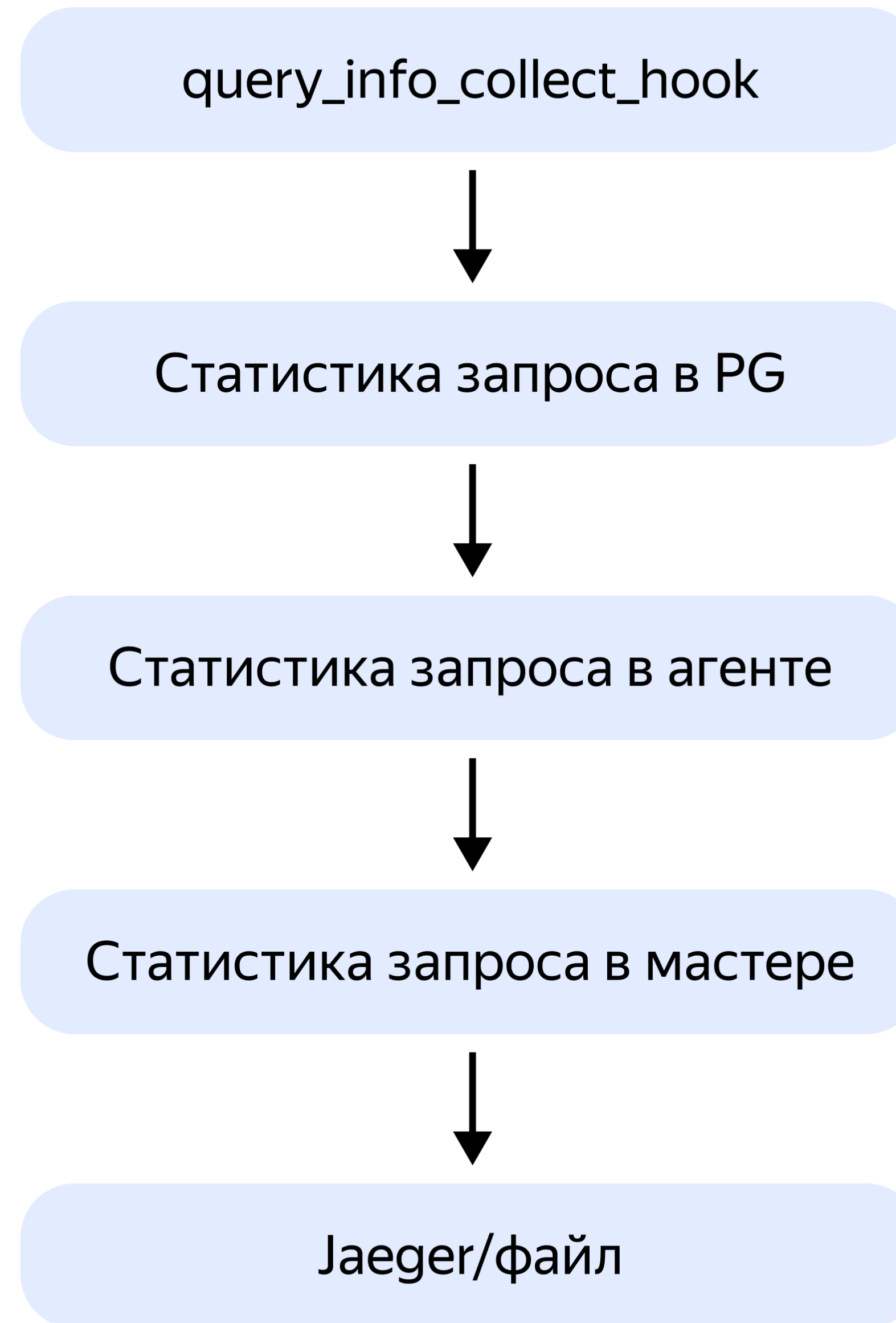
Можно обновлять
без рестарта БД

3

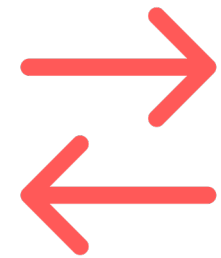
Падение агента
не влияет на БД

Что это даёт

Трассировка сессии/запроса

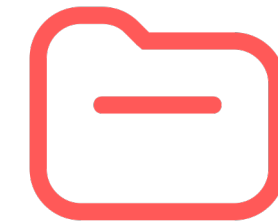


Что мы потеряли



Производительность

1. Отправка данных затратнее обновления структуры в shared памяти
2. Необходимо обрабатывать каждый хук



Память

Текст запроса и план выполнения хранятся в агенте

Хуки — query_info_collect_hook

src/backend/executor/execMain.c:

void

standard_ExecutorStart(QueryDesc *queryDesc, int eflags)

{

 EState *estate;

 MemoryContext oldcontext;

 ...

 /* GPDB hook for collecting query info */

 if (query_info_collect_hook)

 (*query_info_collect_hook)(METRICS_QUERY_START, queryDesc);

Хуки — wait event

src/backend/utils/activity/wait_event.c:

```
void  
pgstat_set_wait_event_storage(uint32 *wait_event_info)  
{  
    my_wait_event_info = wait_event_info;  
}
```


В хуках Executor отправляем статистику в агент

1

На всех этапах используем автобатчинг и дропы данных

2

Неполнота — свойство системы

3

Гибкость и простота обновления

4

ГОТОВ ОТВЕТИТЬ НА ВАШИ ВОПРОСЫ



Леонид Борчук
Team Lead Greenplum,
Yandex Cloud