Serverless под капотом

DOTNEXT ПИТЕР | 16 МАЯ 2019

Михаил Шилков

Я в Питере (2005)

Я в Нидерландах (2013)

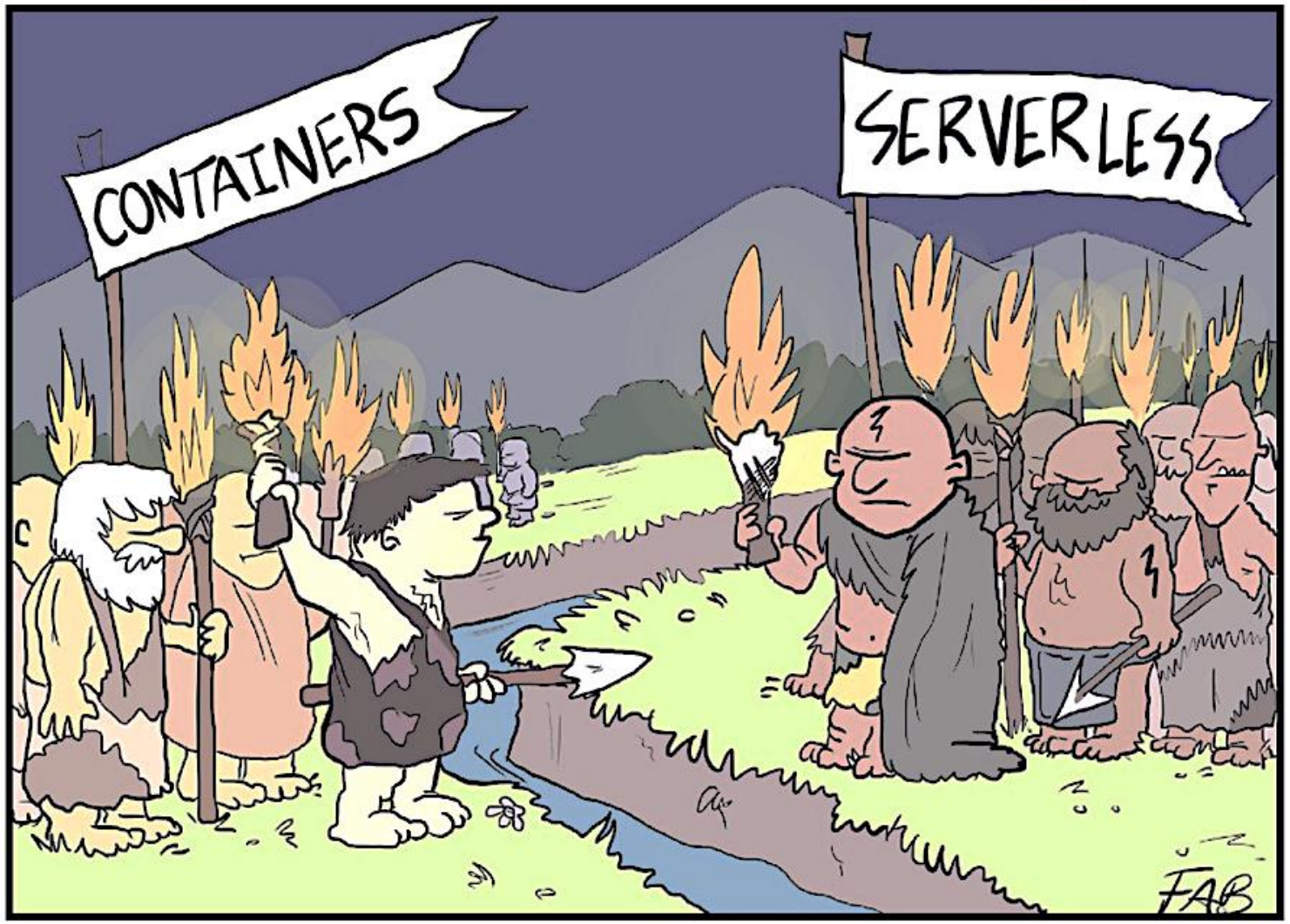
План

Cloud & Serverless

Цели и философия

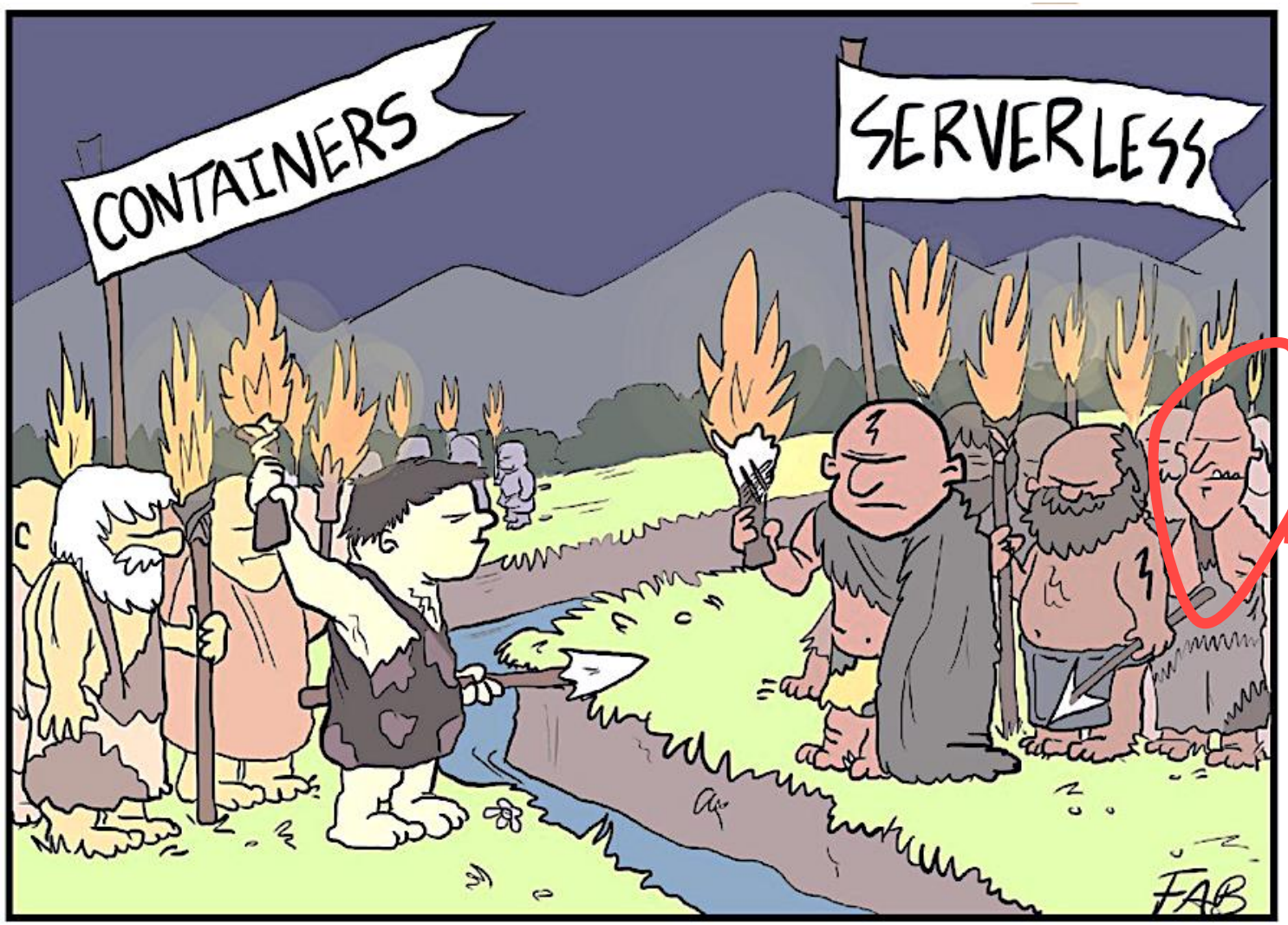
Подходы к реализации

Тесты, графики, выводы



Облачная Архитектура

Рисунок © Forrest Brazeal и A CLOUD GURU



Я :)

Рисунок © Forrest Brazeal и A CLOUD GURU

Рисунок © Forrest Brazeal и A CLOUD GURU

Характеристики Serverless

Серверы – не наша проблема

Масштабируемость и доступность из коробки

Плата за фактическое пользование

Function-as-a-Service

AWS Lambda

Azure Functions

GCP Functions

Возможные сценарии

Таймер

Запускается раз в день, выполняет PowerShell скрипт за 5 минут

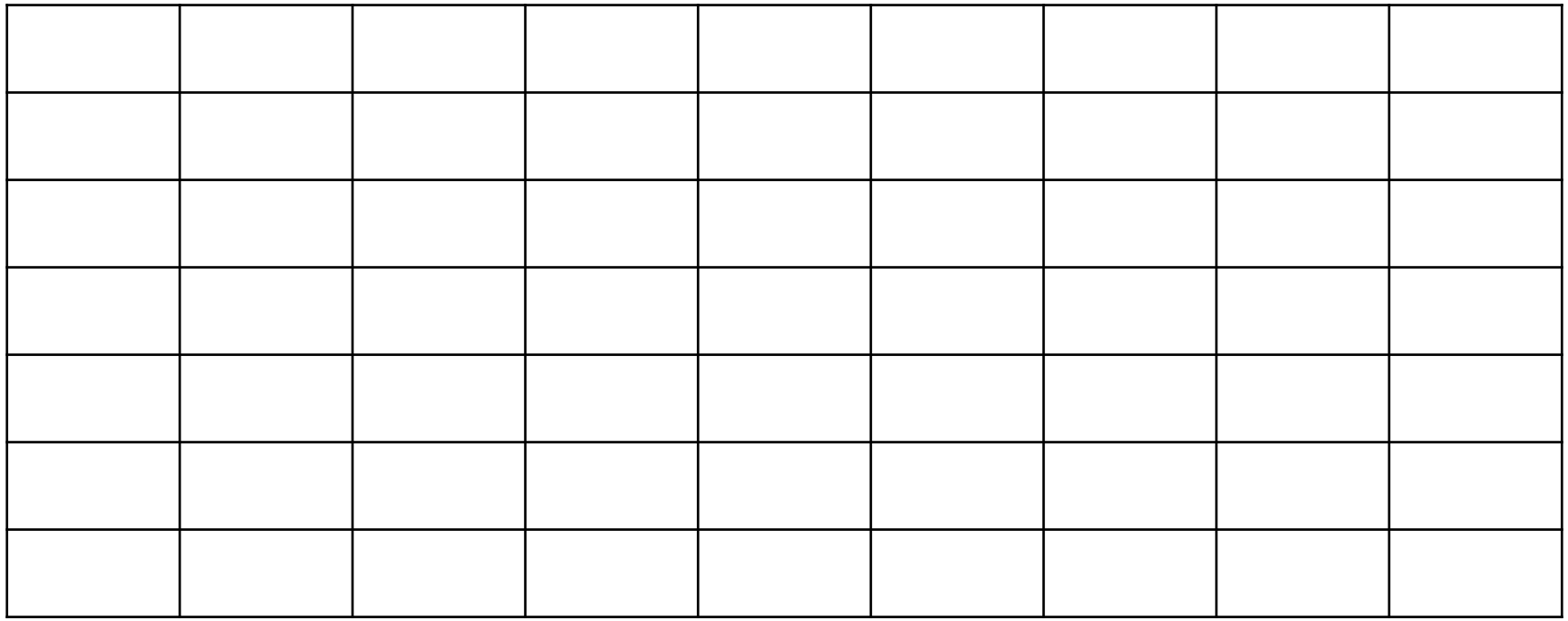
HTTP

Высонагруженный сайт с сотнями запросов в секунду

Очередь

Всплески входящих сообщений, различная длительность обработки

Распределение нагрузки в пуле ресурсов



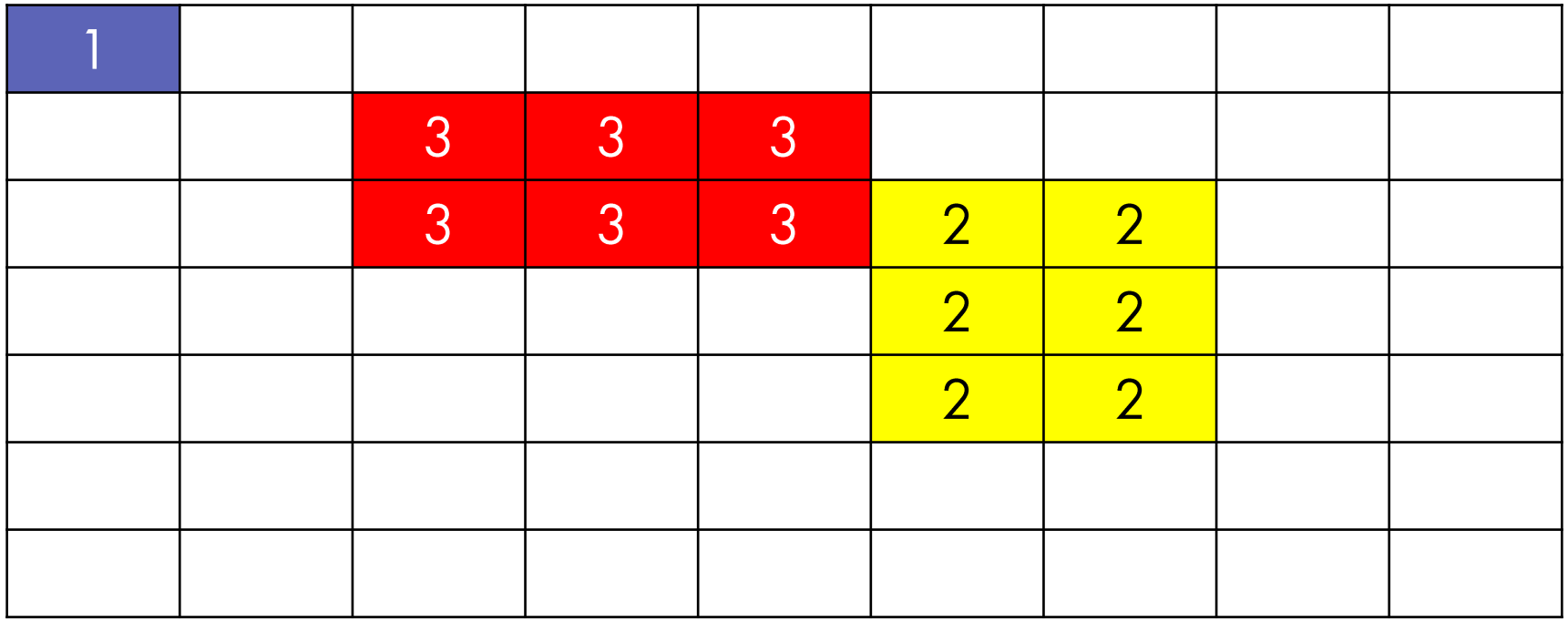
Распределение нагрузки в пуле ресурсов

1								

Распределение нагрузки в пуле ресурсов

1								
					2	2		
					2	2		
					2	2		

Распределение нагрузки в пуле ресурсов



Распределение нагрузки в пуле ресурсов

1								
		3	3	3				
		3	3	3	2	2		
4	4				2	2		
4	4				2	2		
4	4							
4	4							

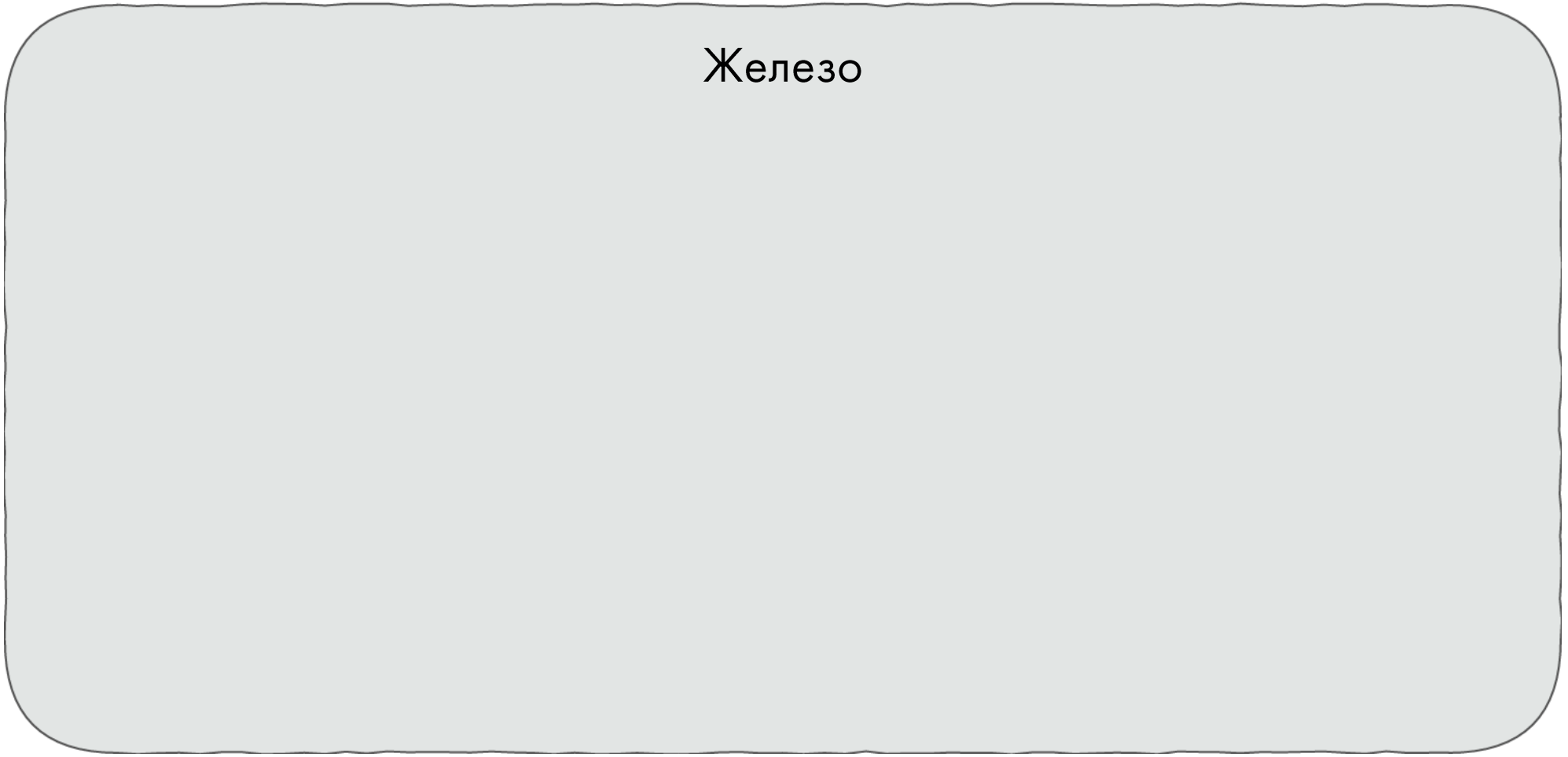
Распределение нагрузки в пуле ресурсов

1								
		3	3	3				
		3	3	3	2	2		
4	4				2	2		
4	4				2	2		
4	4	5	5	5	5	5		
4	4	5	5	5	5	5		

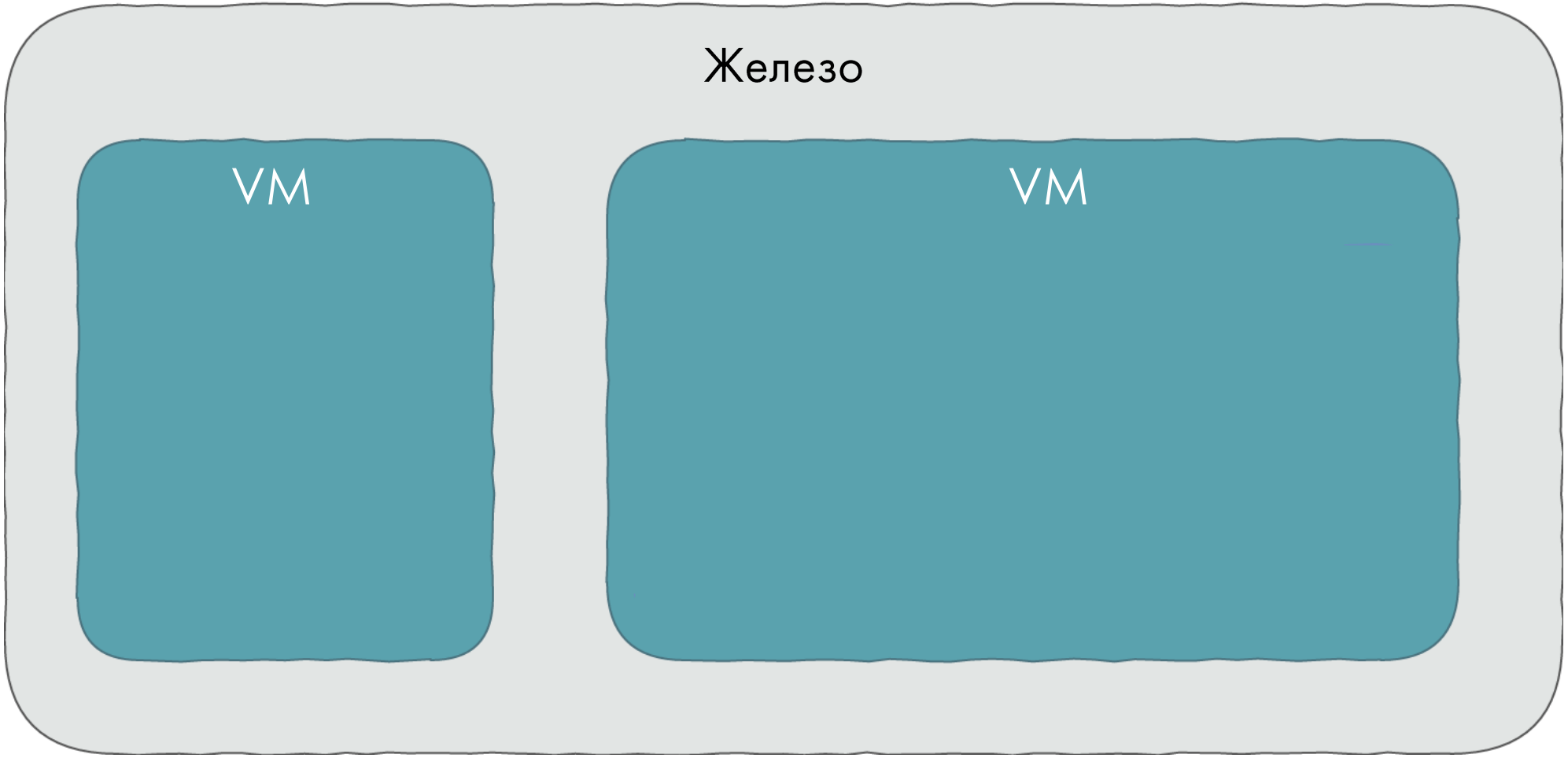
Распределение нагрузки в пуле ресурсов

1								
		3	3	3			6	6
		3	3	3	2	2	6	6
4	4				2	2	6	6
4	4				2	2	6	6
4	4	5	5	5	5	5	6	6
4	4	5	5	5	5	5	6	6

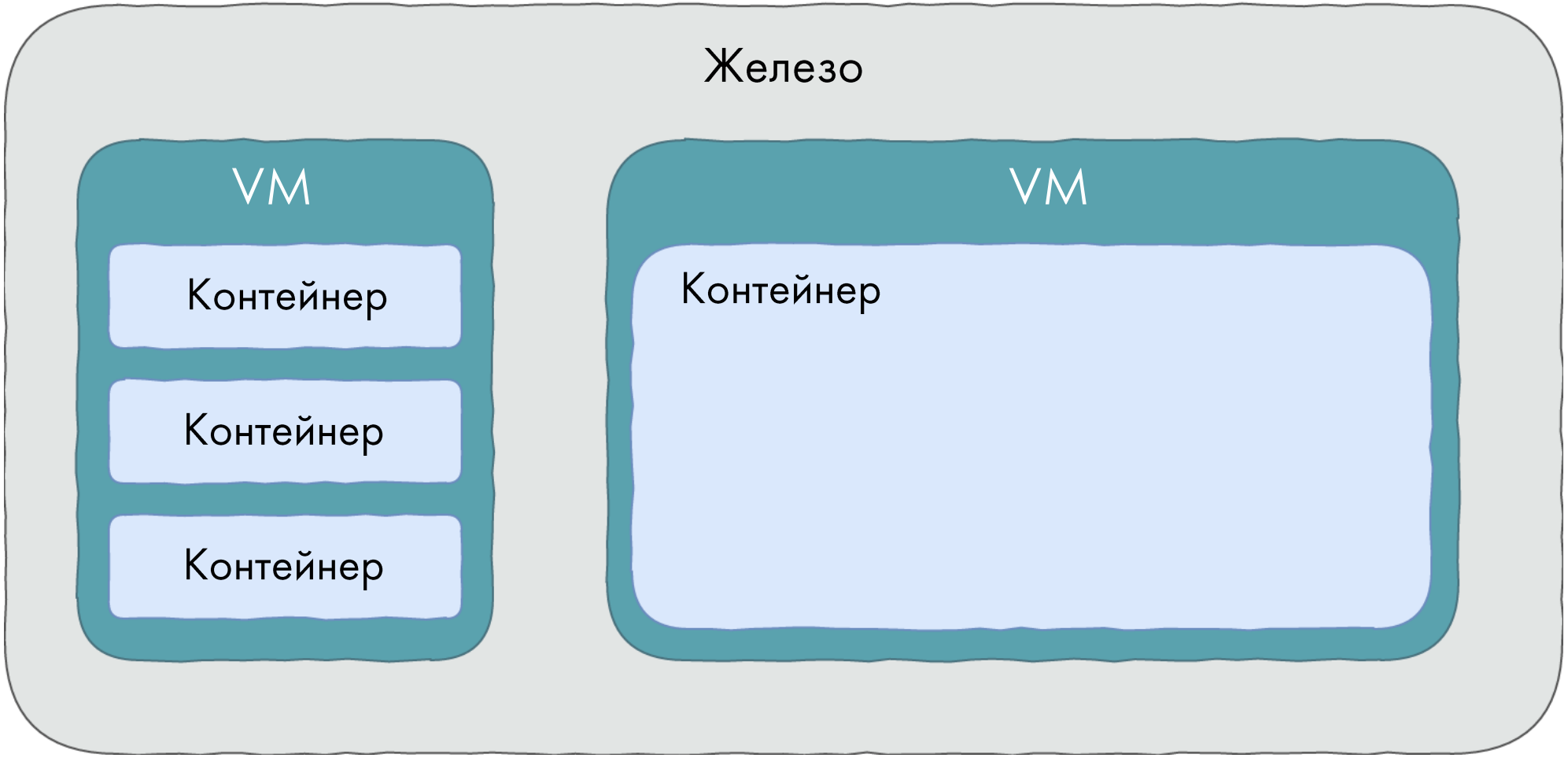
Сервер, VM, контейнер, процесс, ...



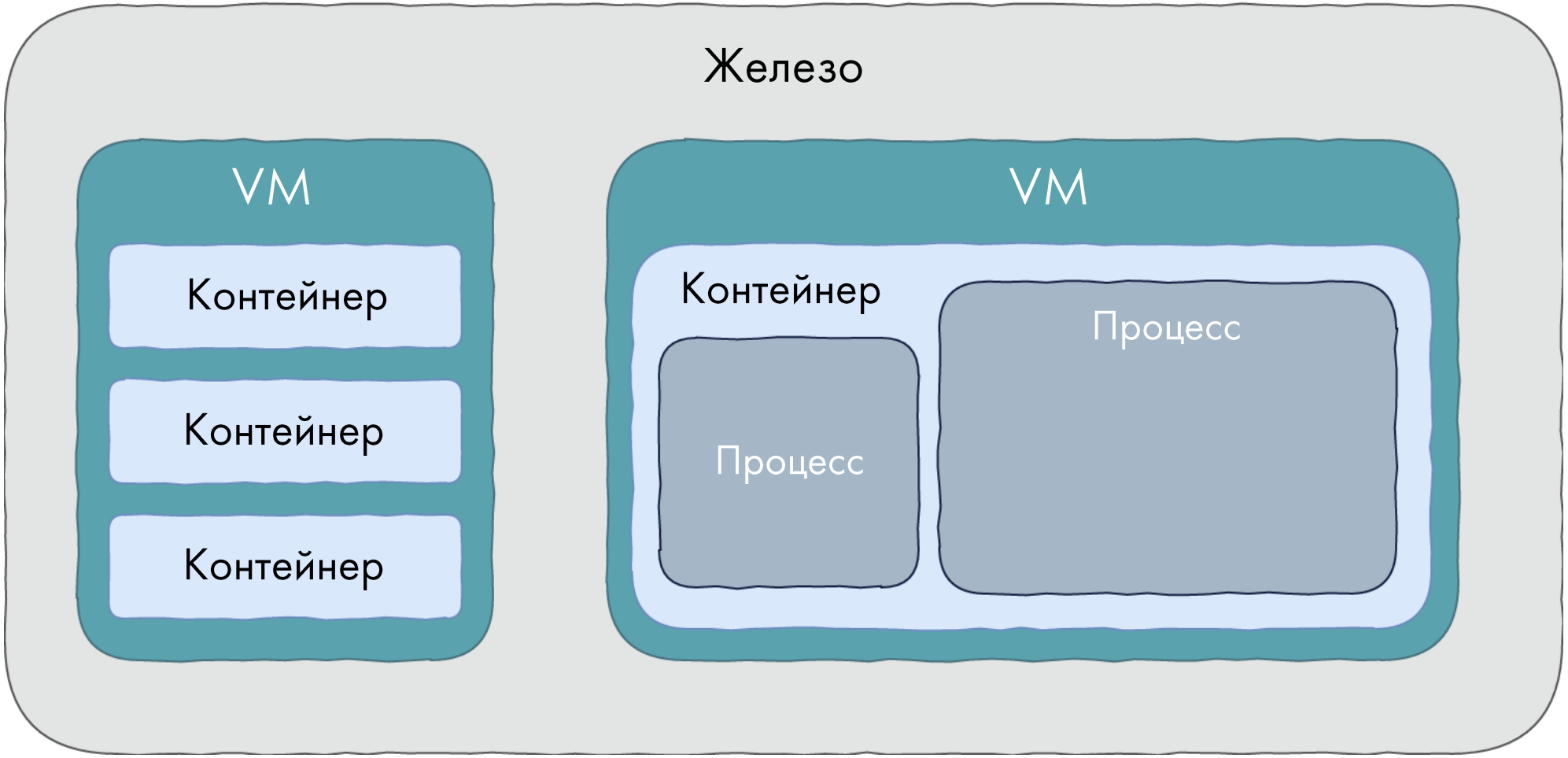
Сервер, VM, контейнер, процесс, ...



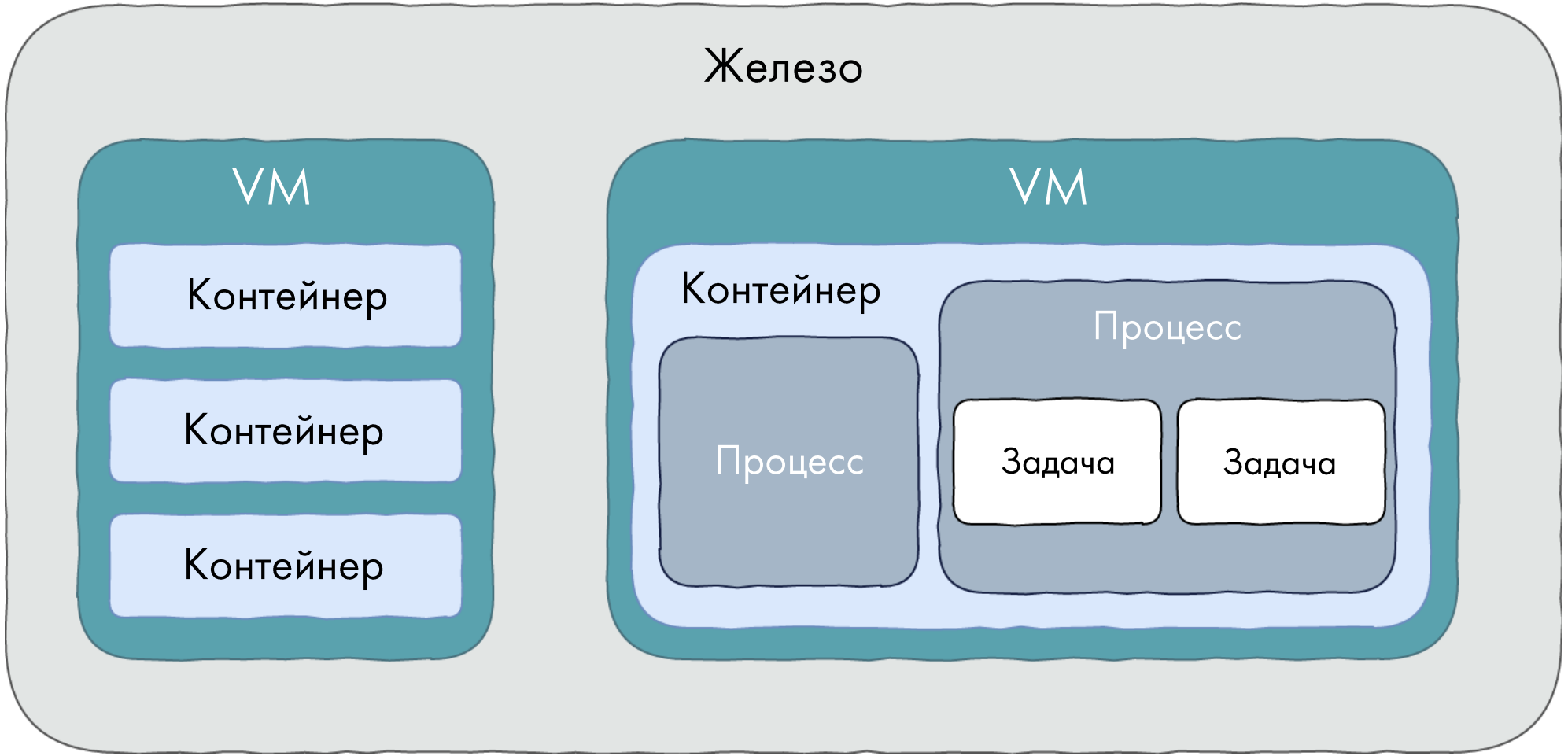
Сервер, VM, контейнер, процесс, ...



Сервер, VM, контейнер, процесс, ...



Сервер, VM, контейнер, процесс, ...

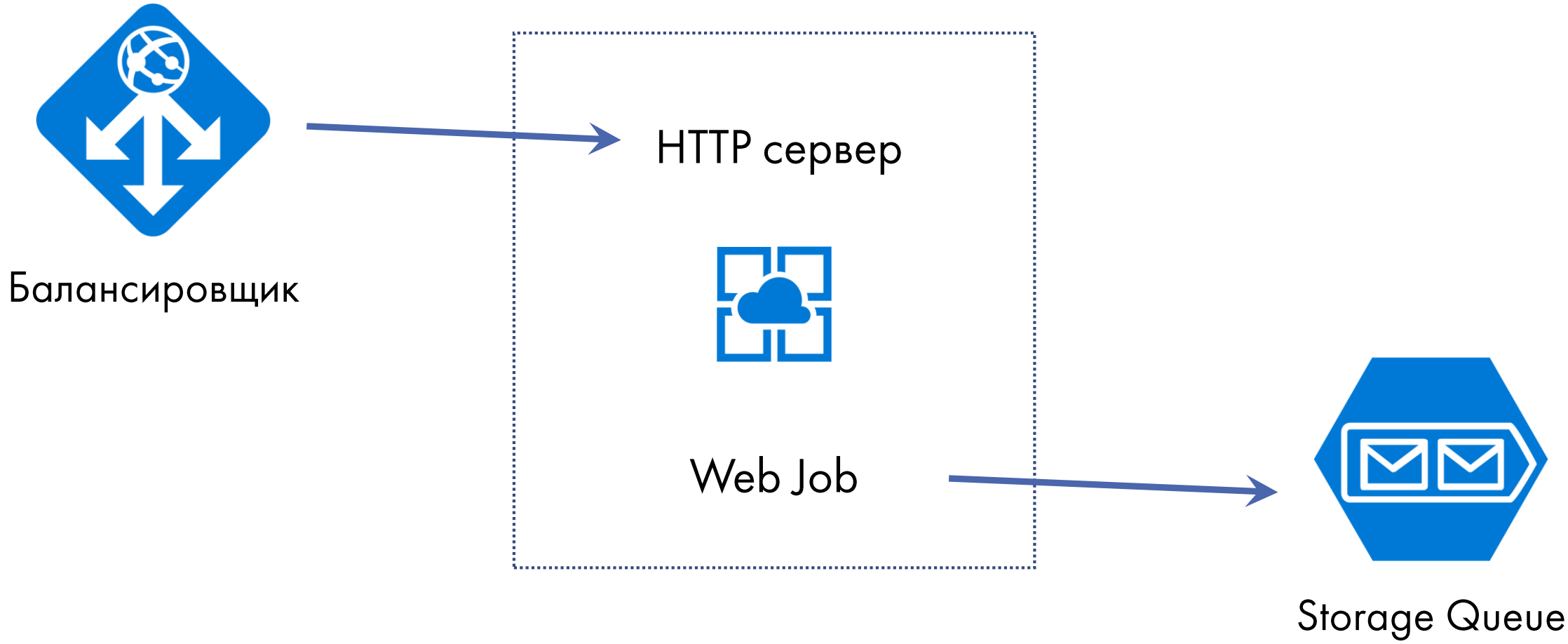


Ценность для Клиента

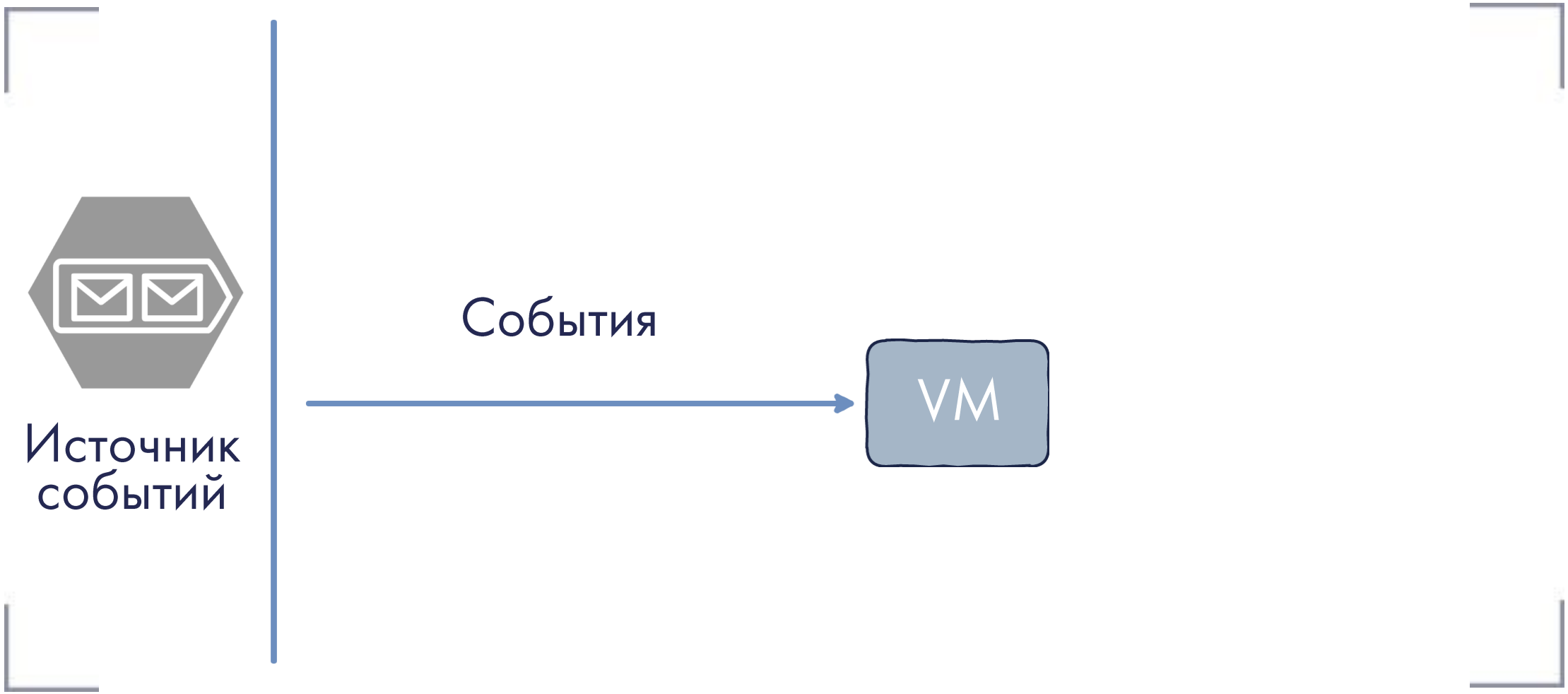
Использование ресурсов

Azure Functions

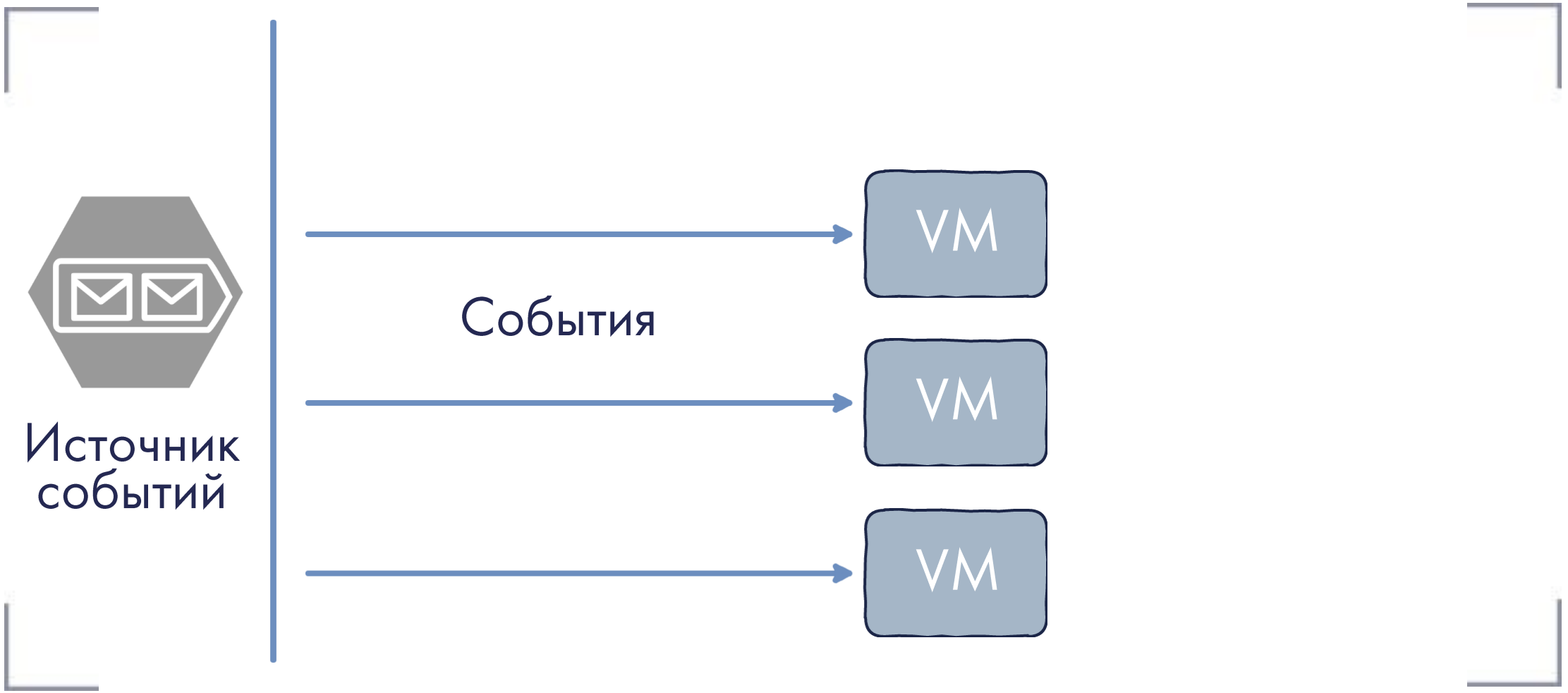
Наследие App Service



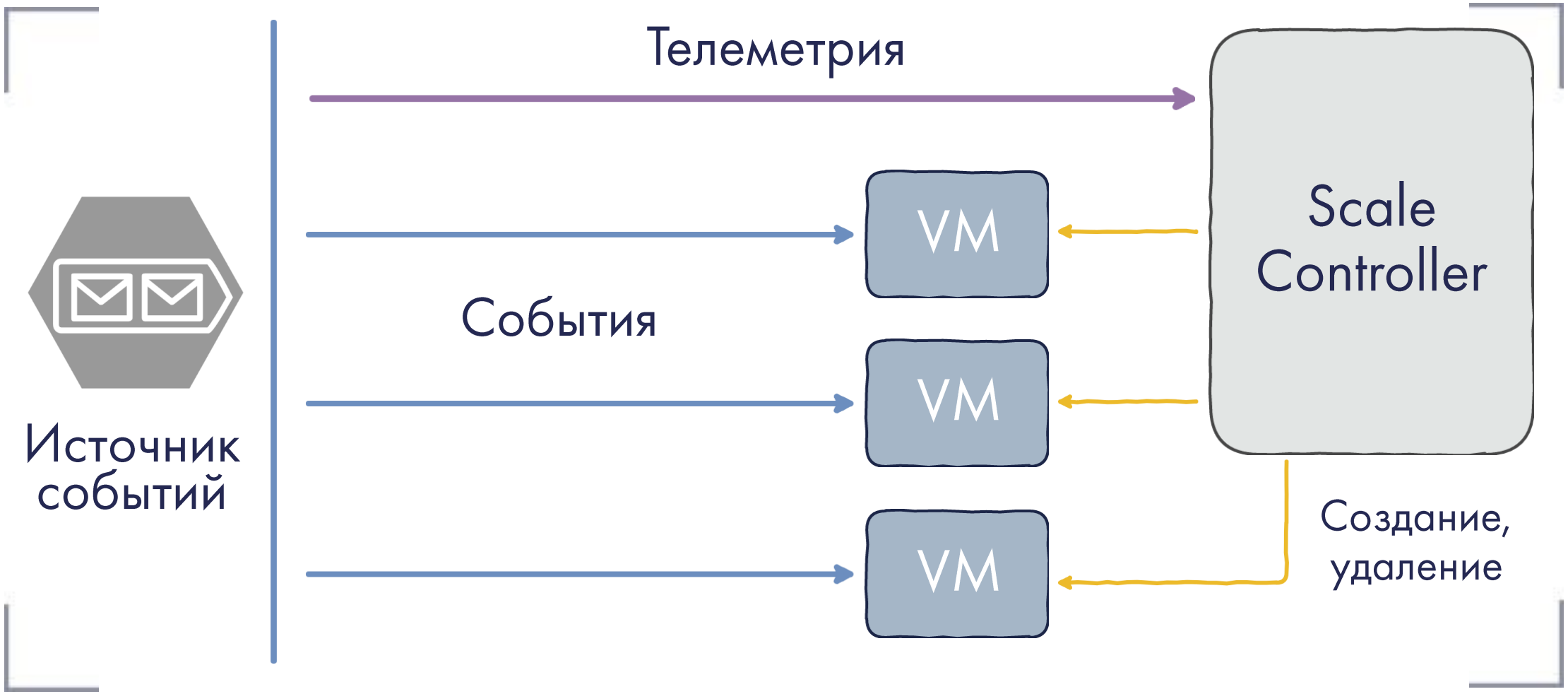
Наследие App Service



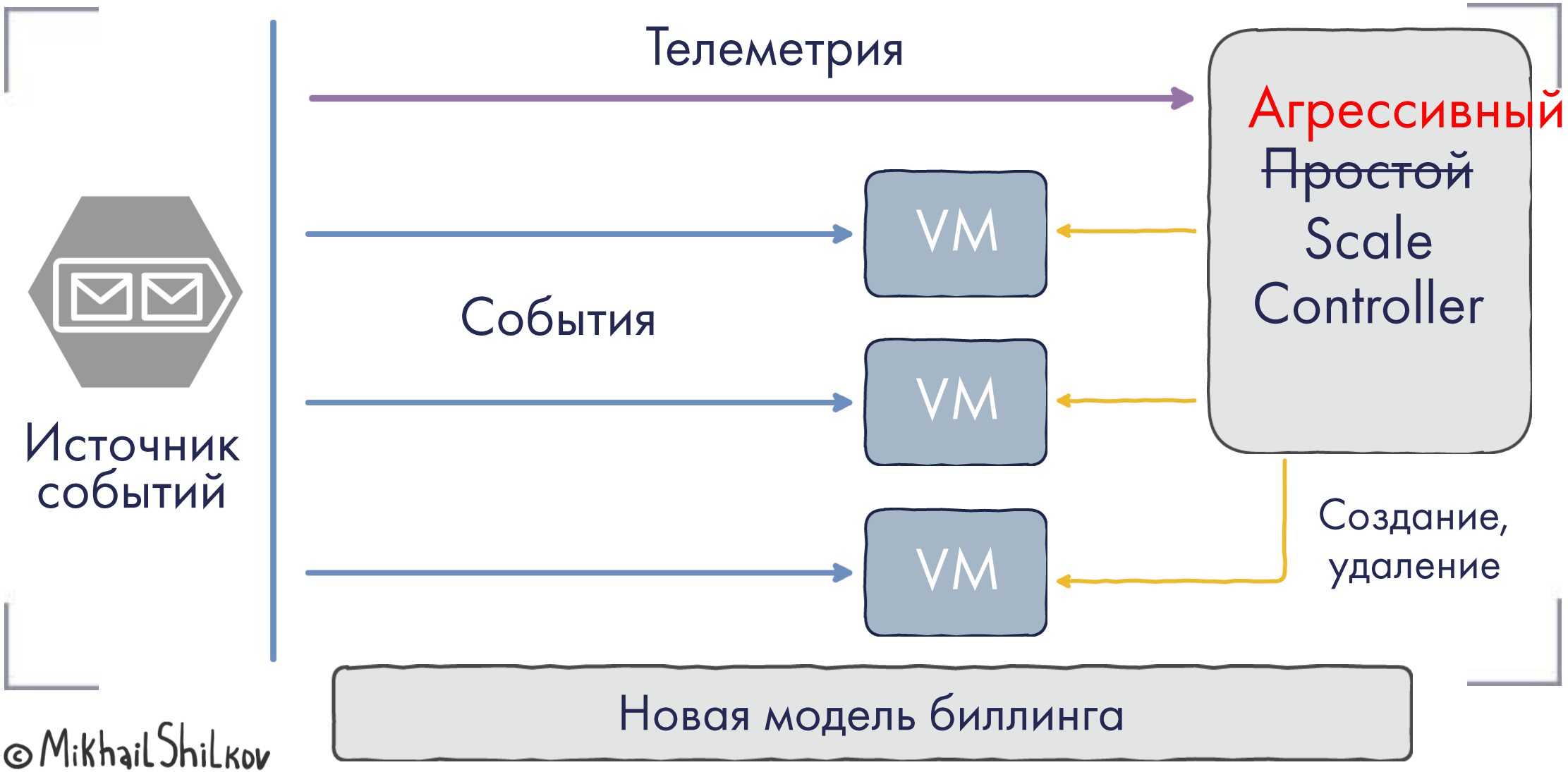
App Service: масштабируемость



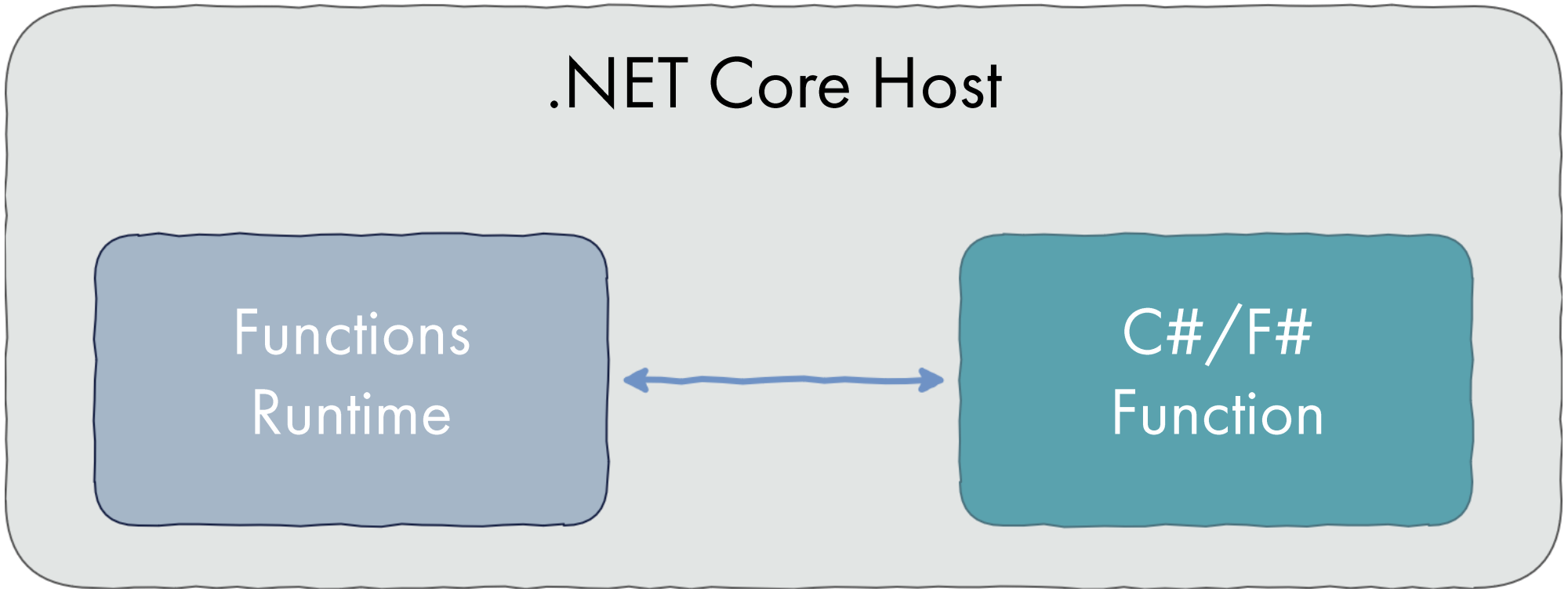
App Service: масштабируемость



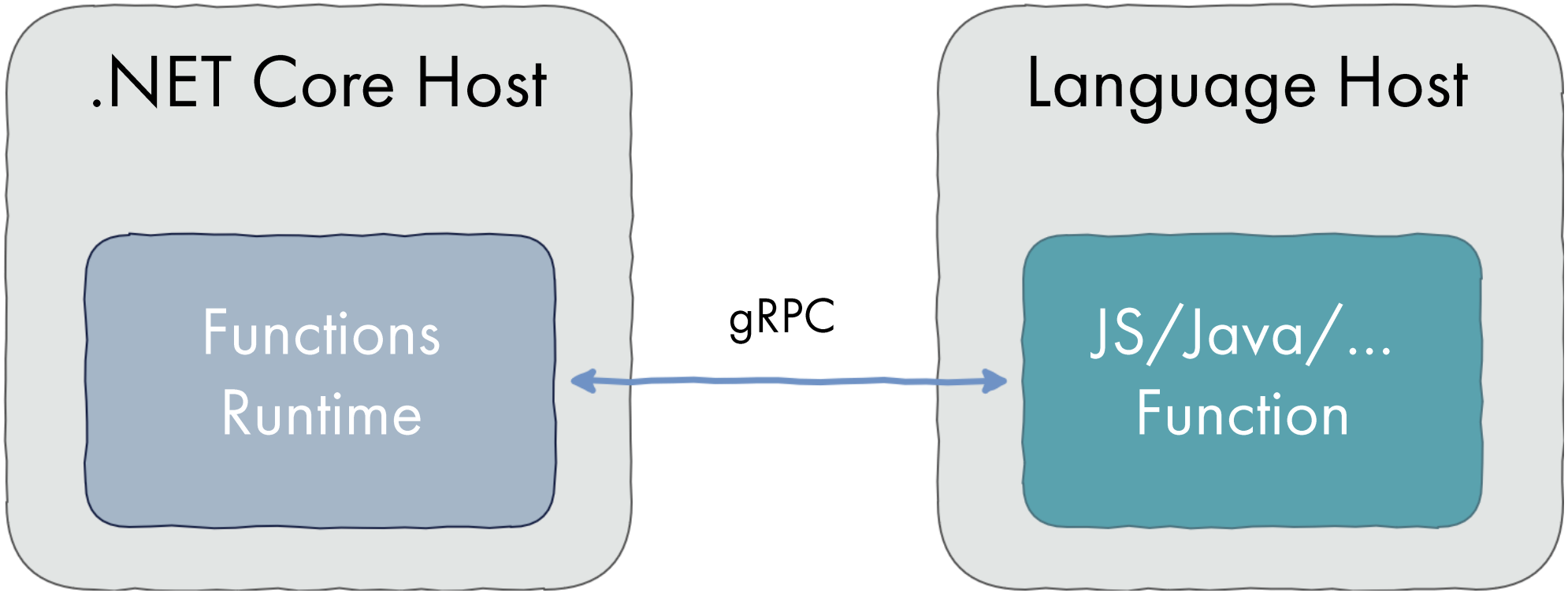
Function App Consumption Plan



.NET в Azure Functions (in-proc)



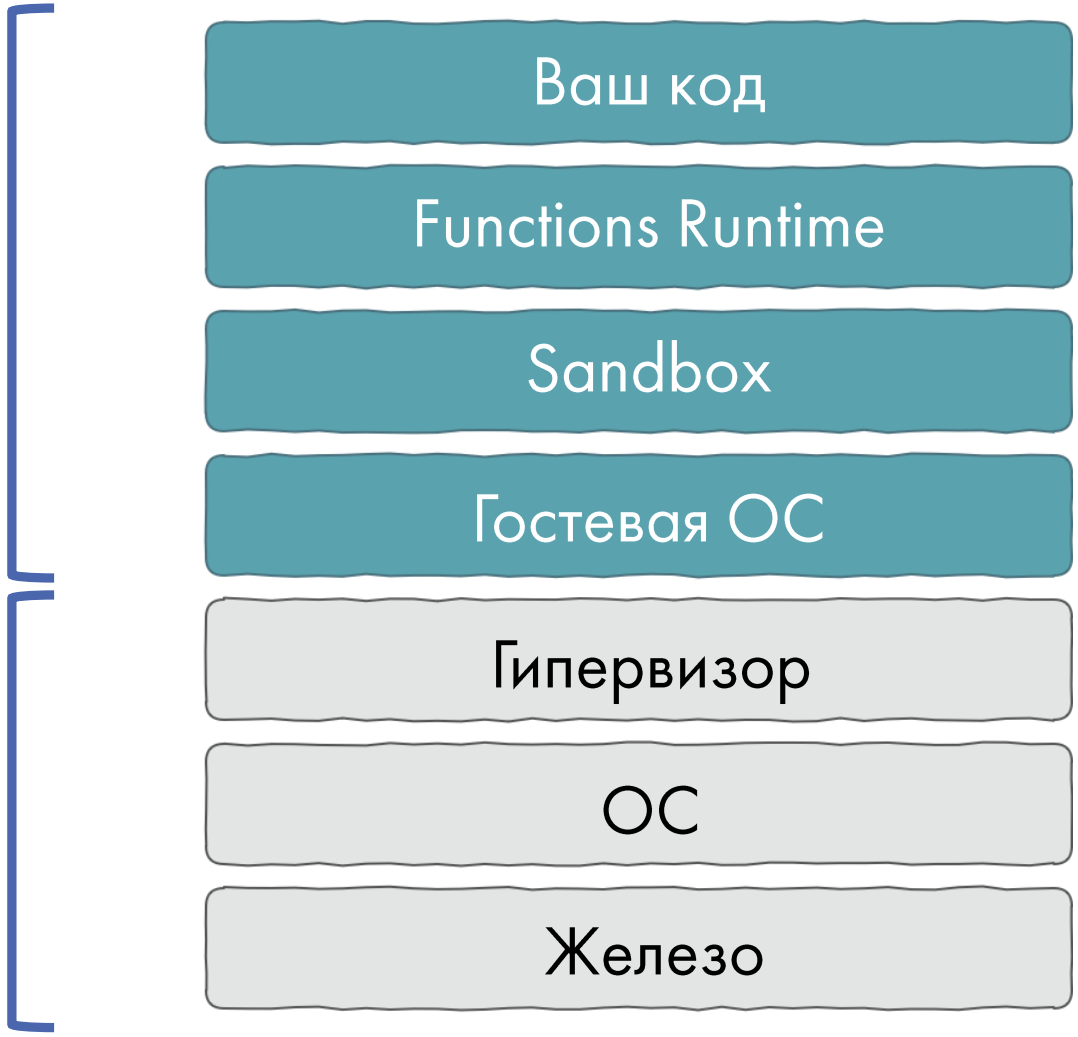
.NET в Azure Functions (out-of-proc)



Уровни изоляции

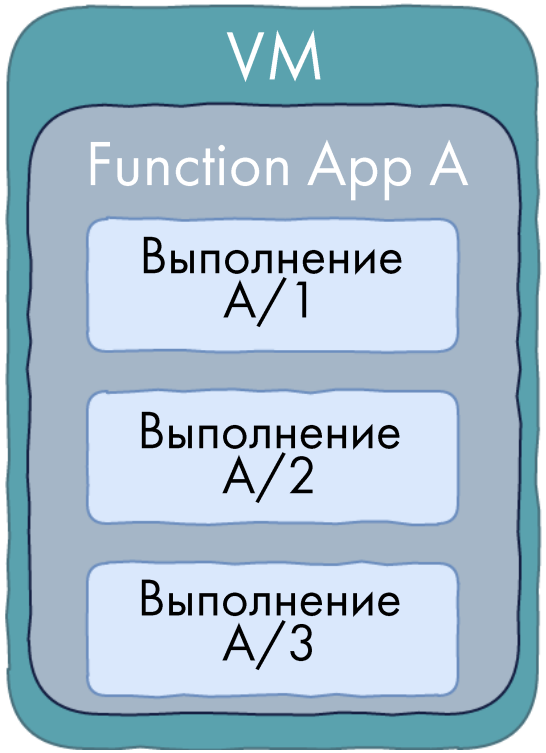
Одно приложение (VM)

Разные приложения

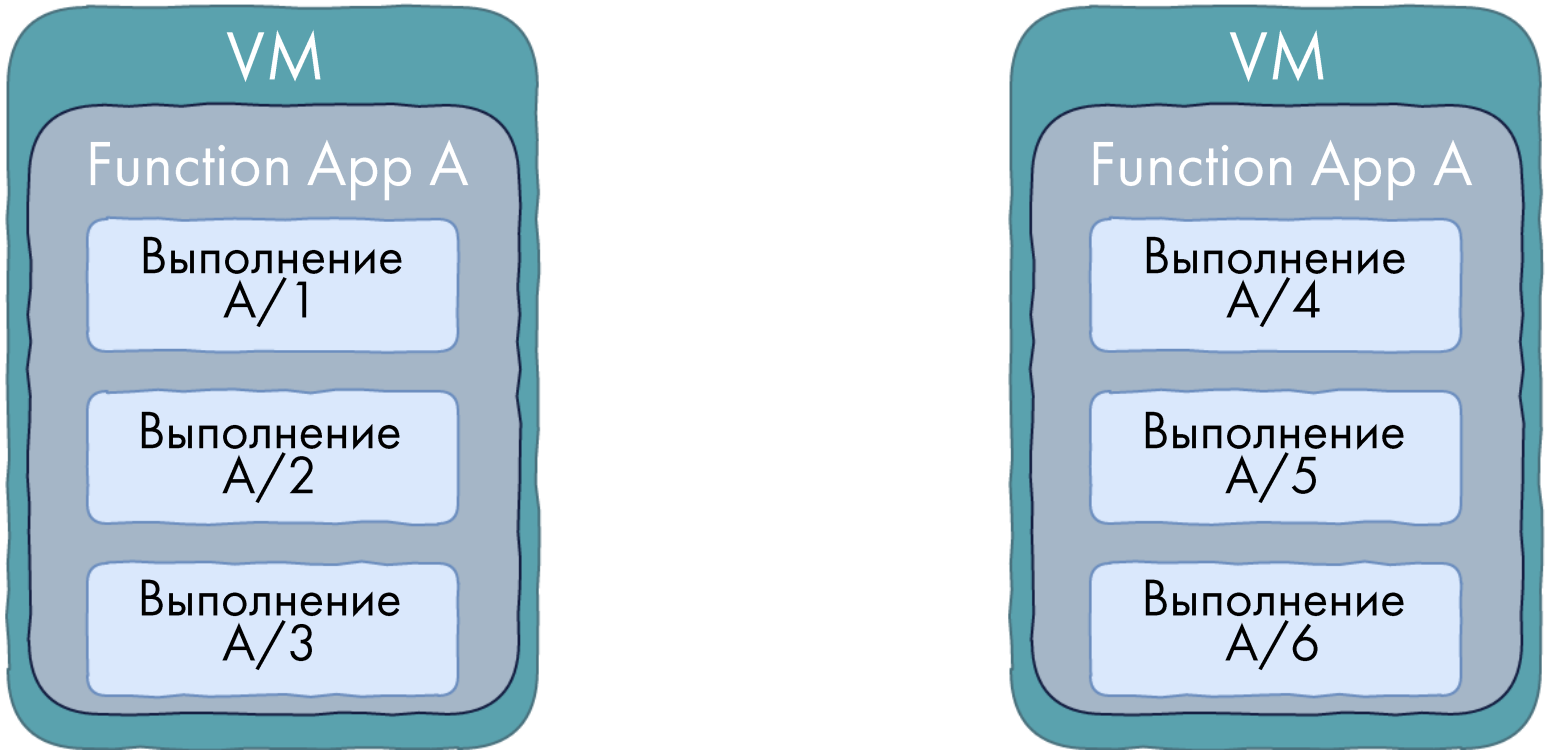


Одновременные выполнения

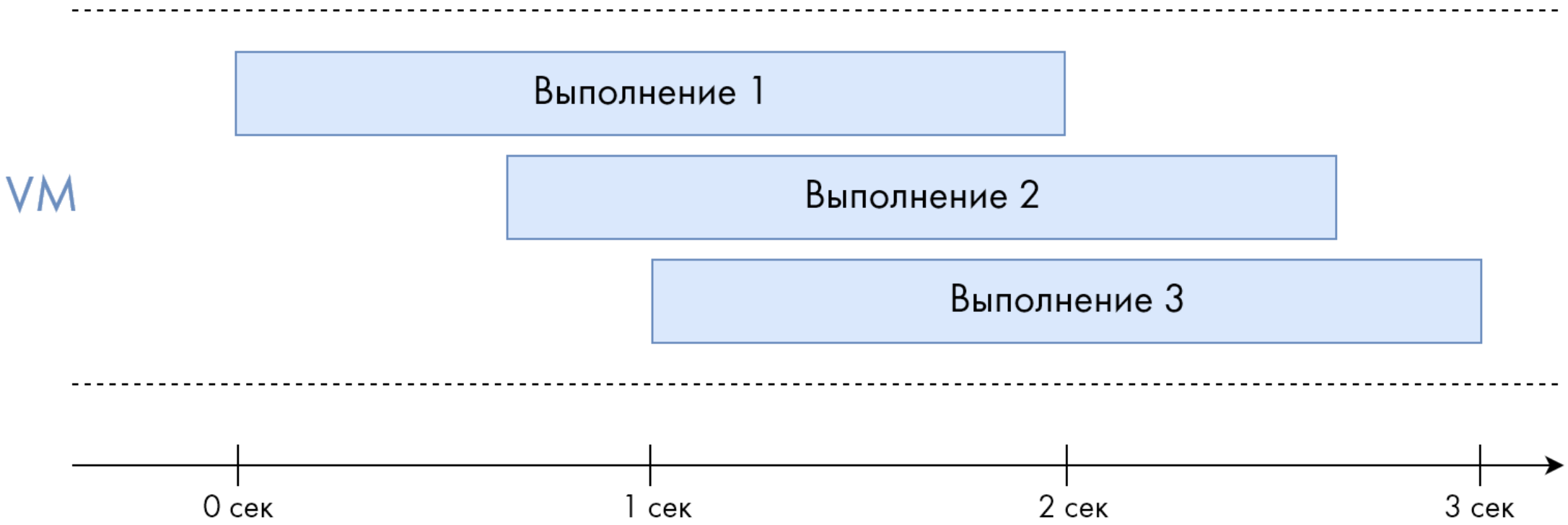
Одновременные выполнения



Одновременные выполнения



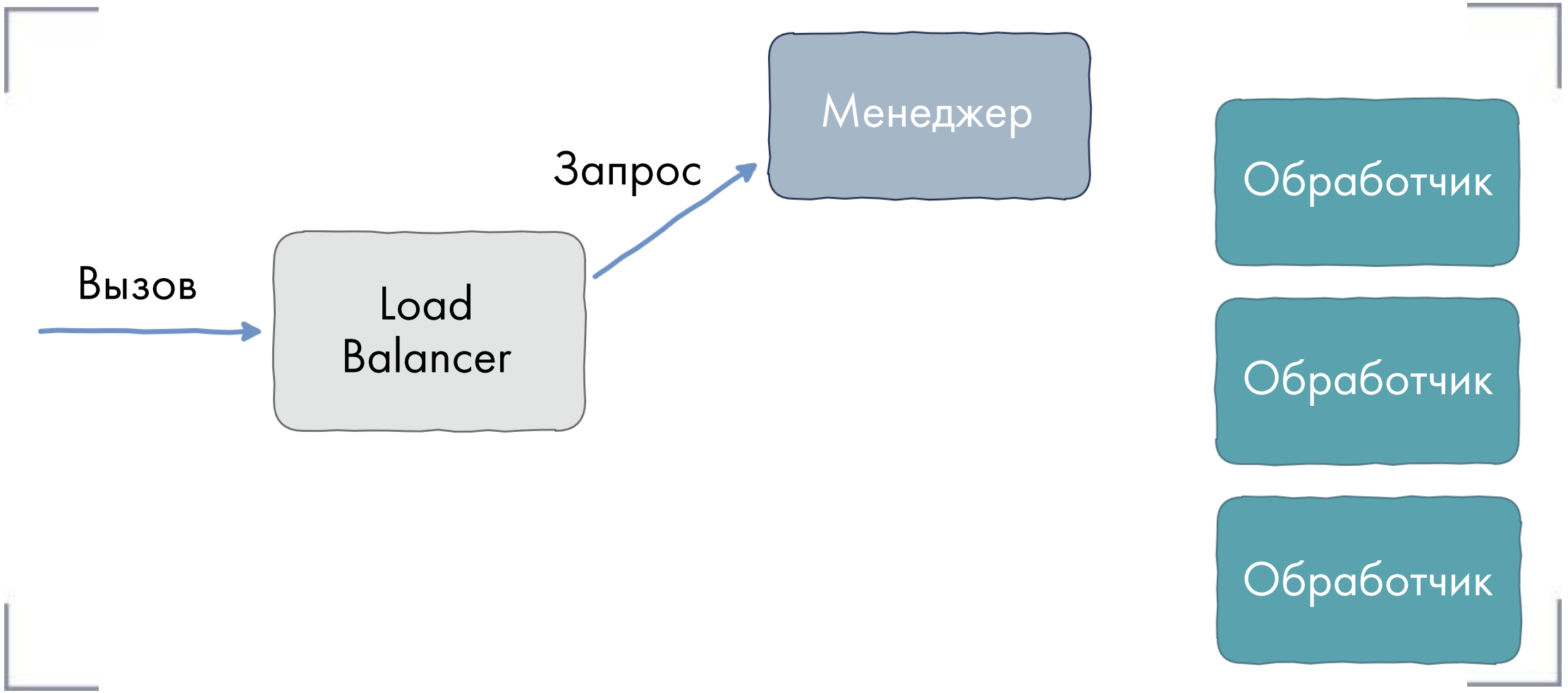
Одновременные выполнения



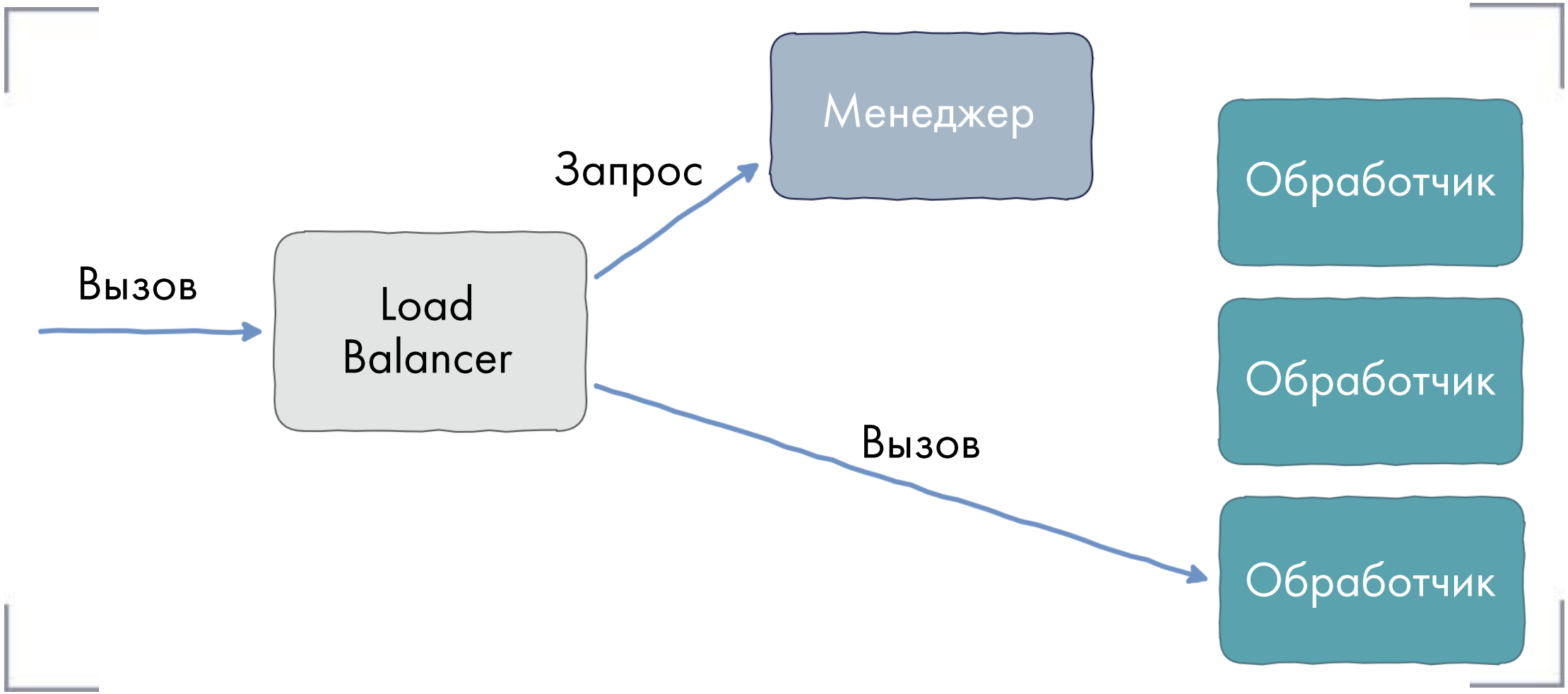
AWS Lambda

AWS Lambda

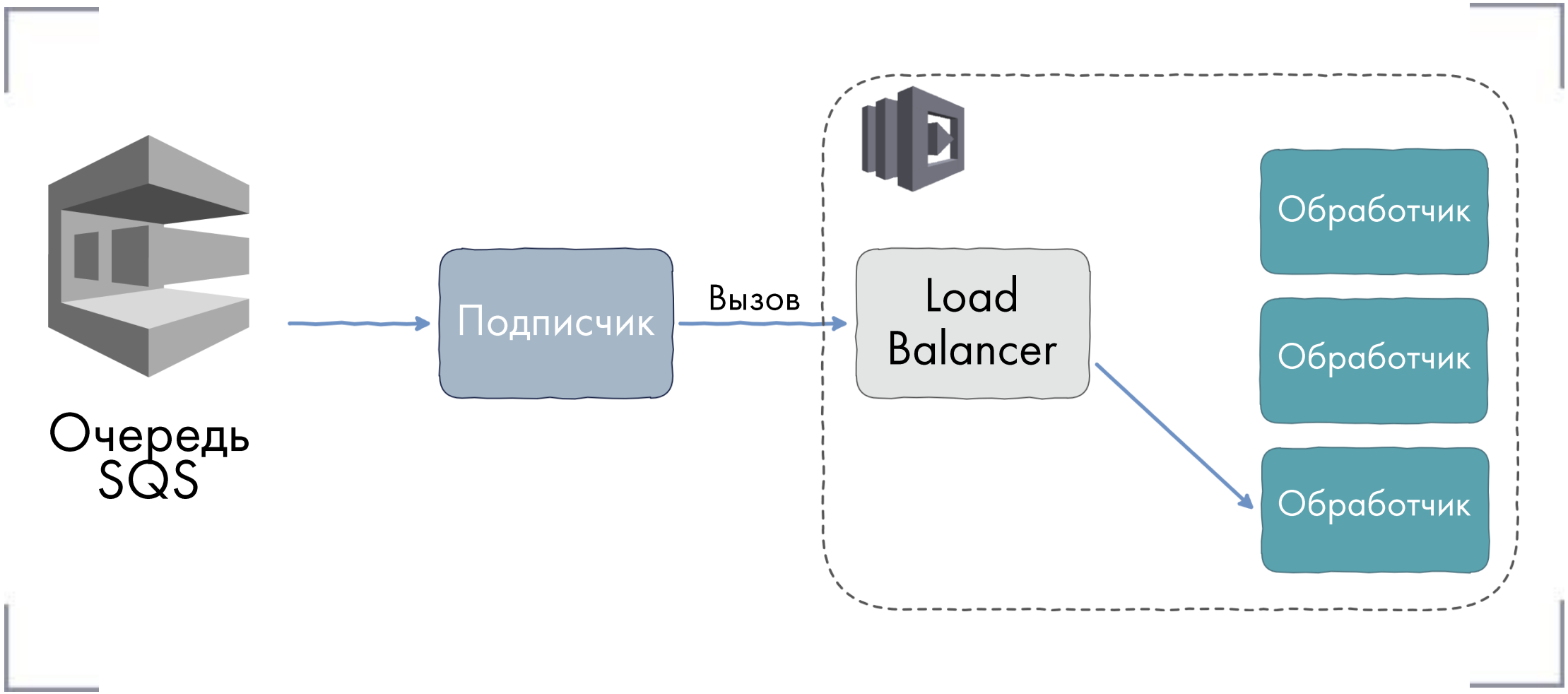
AWS Lambda



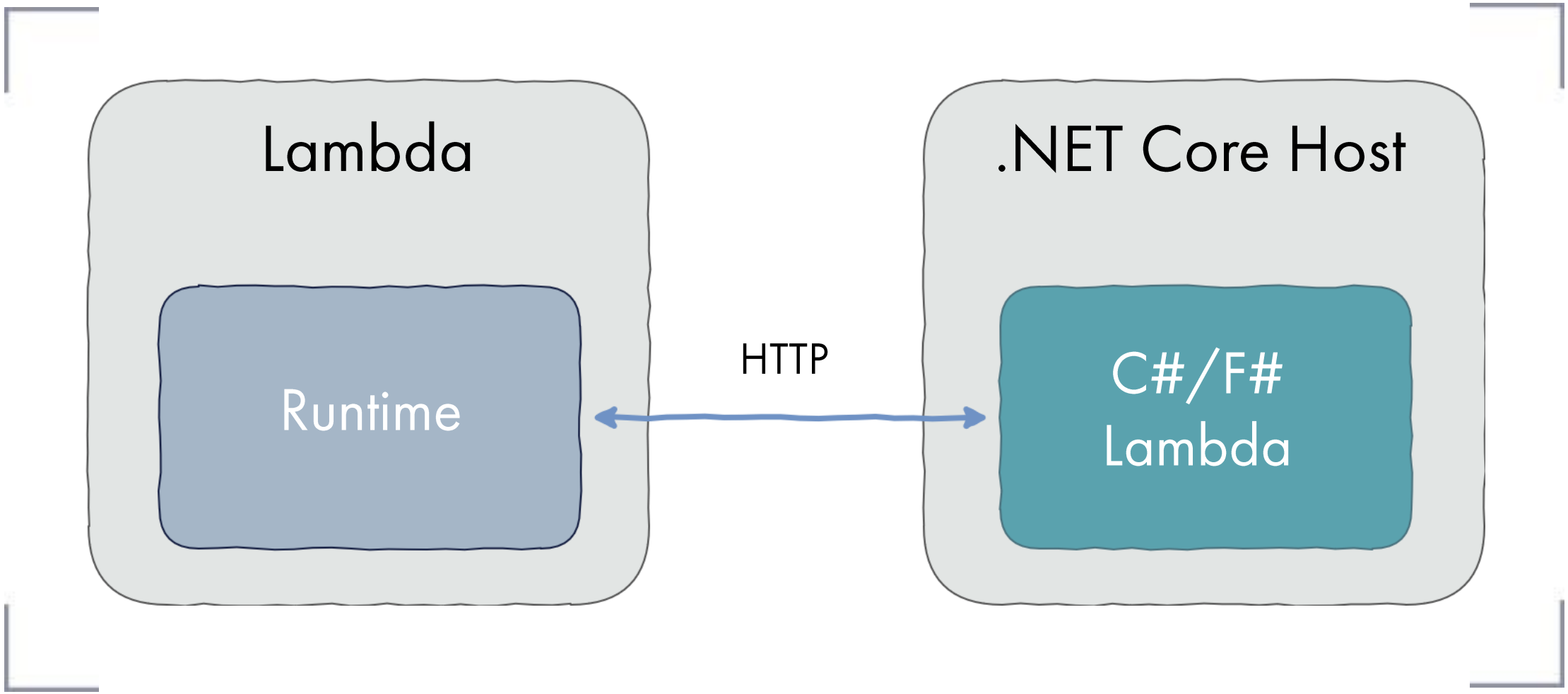
AWS Lambda



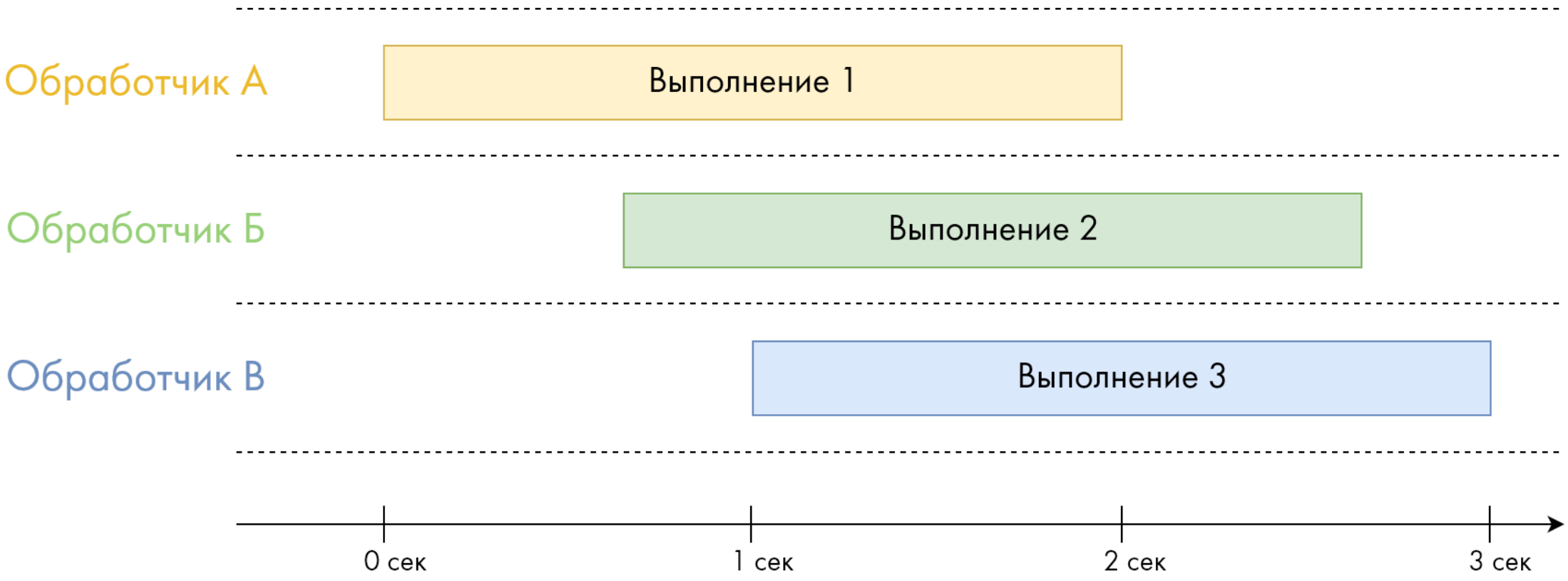
Интеграция с AWS Lambda



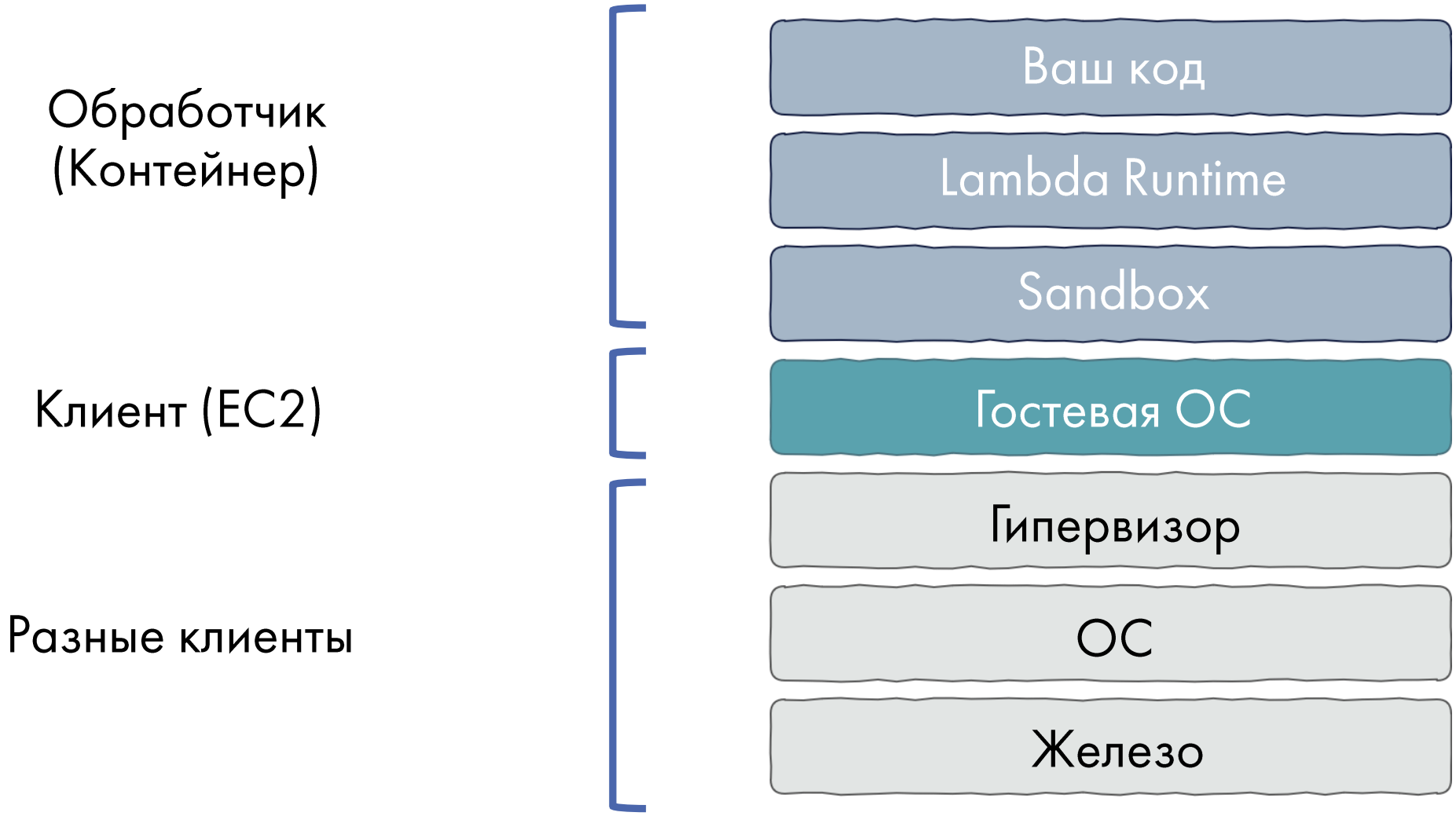
.NET в AWS Lambda



AWS: одновременные выполнения



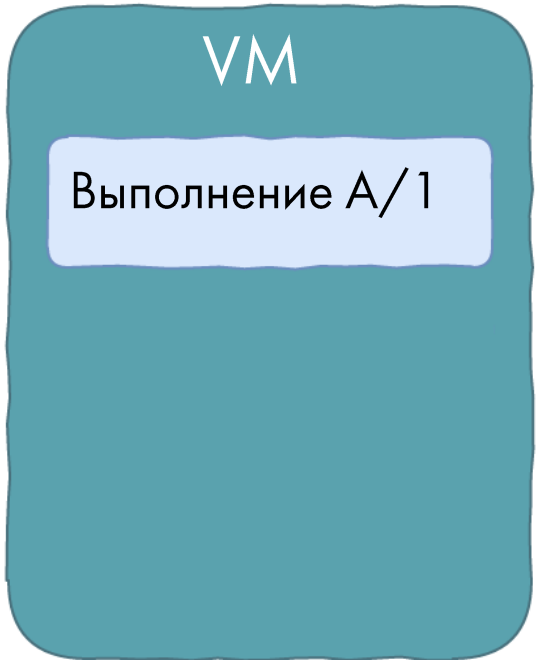
Уровни изоляции



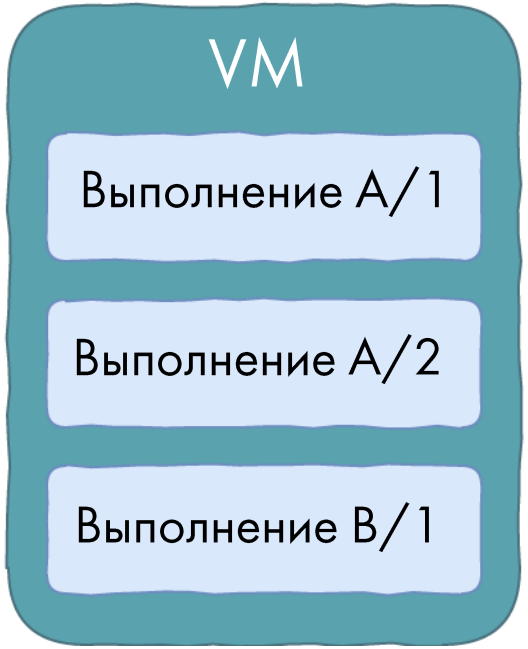
Память (МБ)	Объем бесплатного времени (секунд в месяц)	Цена за 100 мс (USD)
128	3 200 000	0,000000208
192	2 133 333	0,000000313
256	1 600 000	0,000000417
320	1 280 000	0,000000521
384	1 066 667	0,000000625
448	914 286	0,000000729
512	800 000	0,000000834
576	711 111	0,000000938
640	640 000	0,000001042
704	581 818	0,000001146
768	533 333	0,000001250

Память и CPU

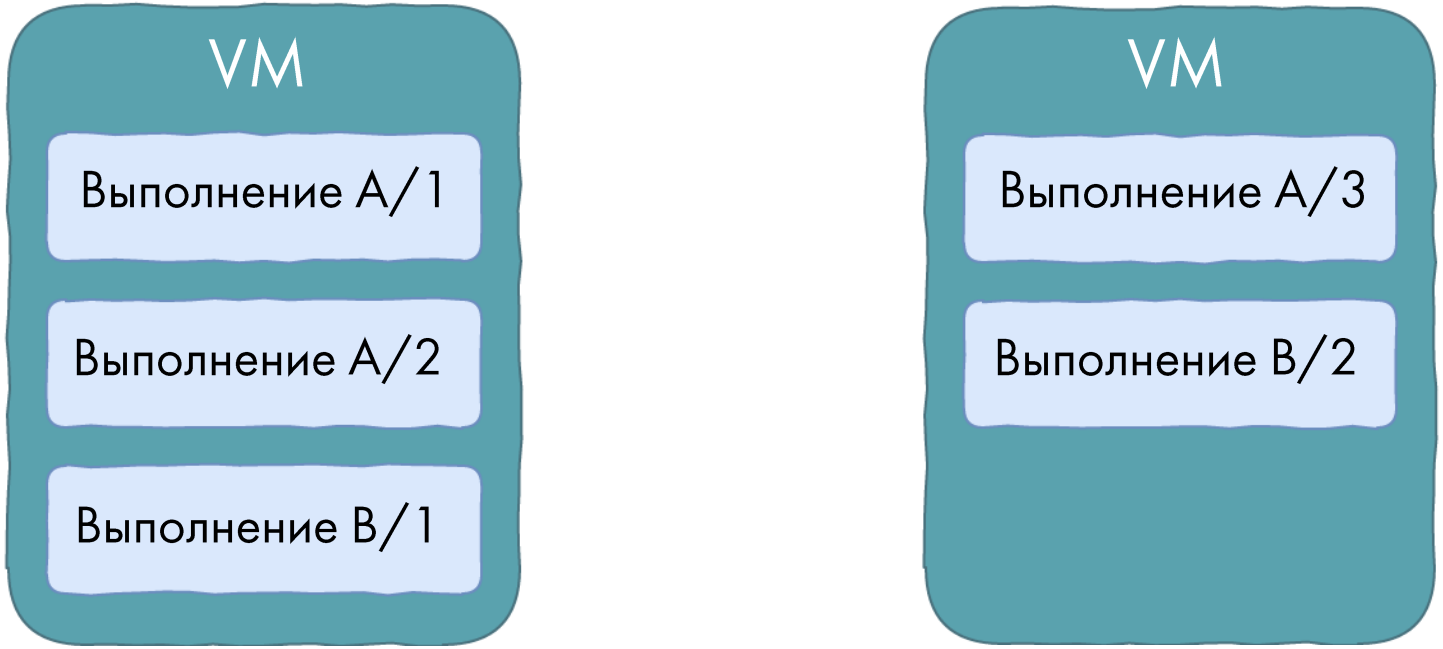
AWS: одновременные выполнения



AWS: одновременные выполнения



AWS: одновременные выполнения



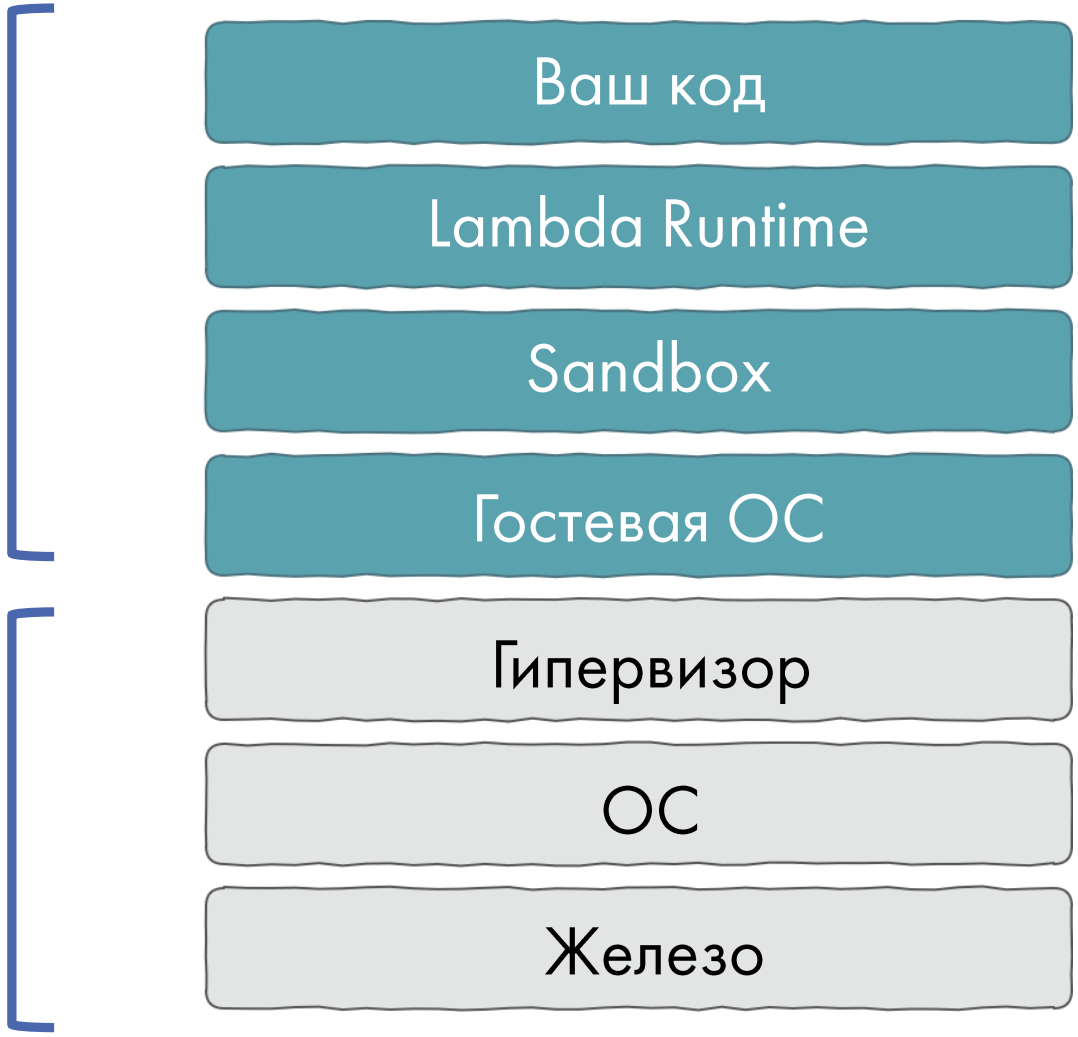
- Микро виртуальная машина
- Загрузка <125мс, 5 МБ расход памяти

- AWS Lambda и Fargate
- Тысячи VM на одном сервере

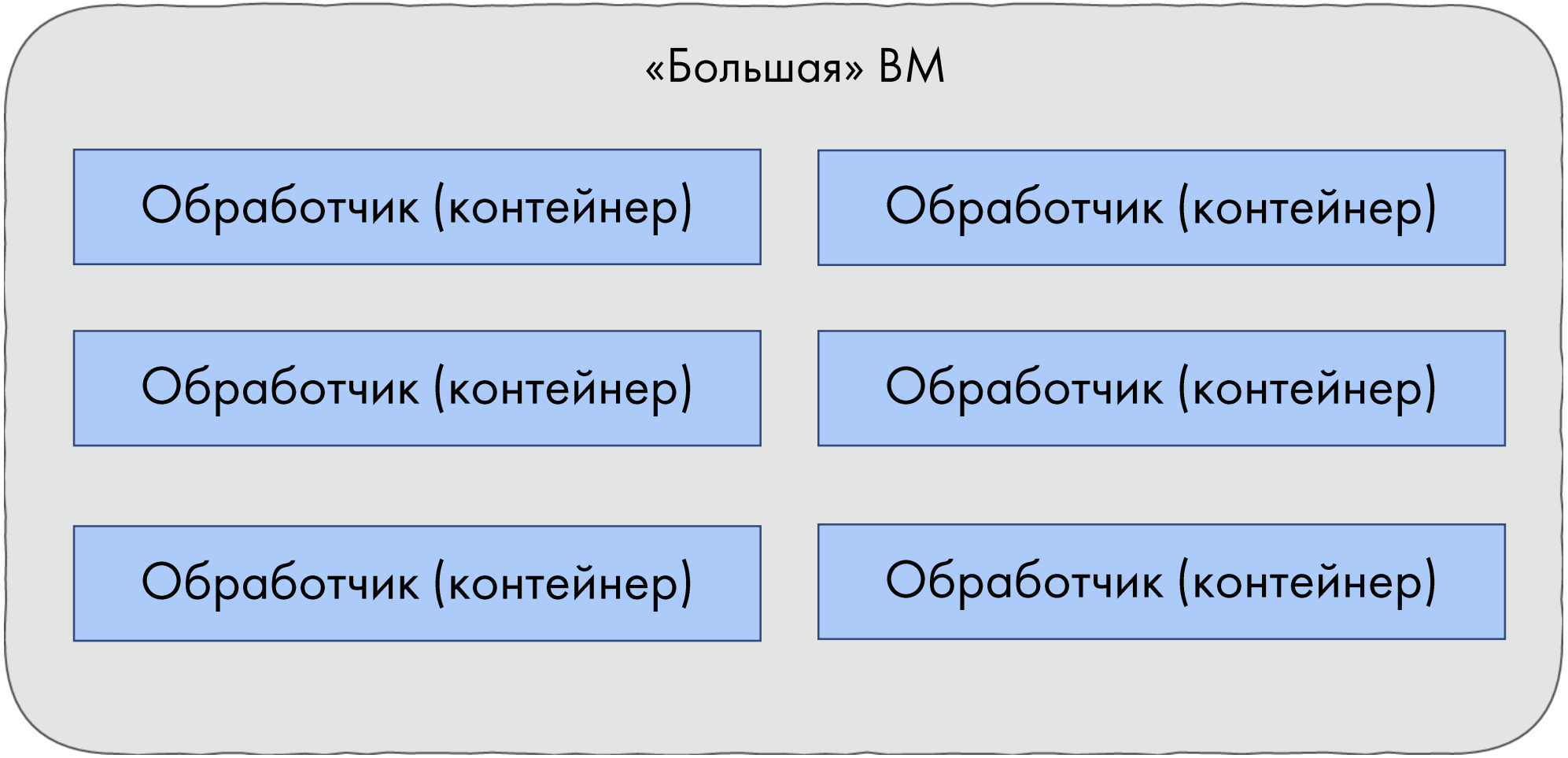
Уровни изоляции с Firecracker

Один обработчик
(Микро VM)

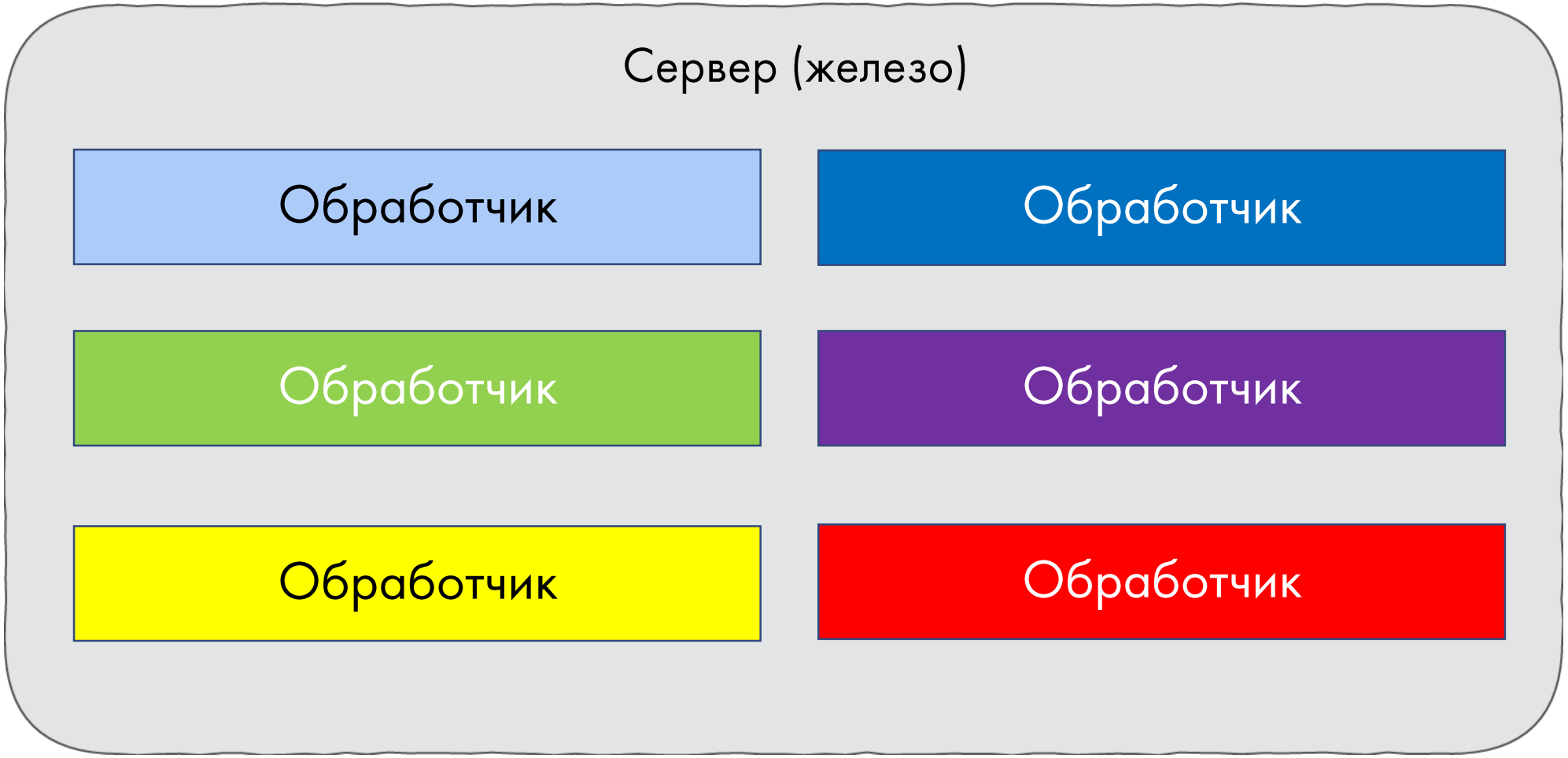
Разные клиенты



«Наивная» компоновка функций



Статистическое мультиплексирование

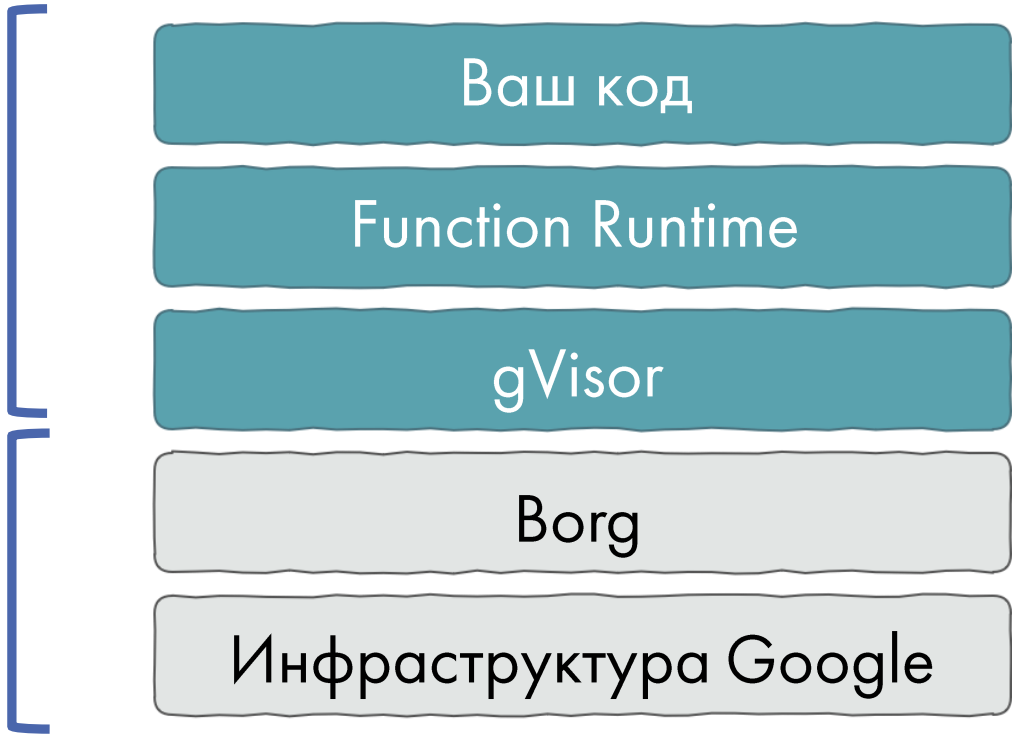


Google Cloud Functions & Cloudflare Workers

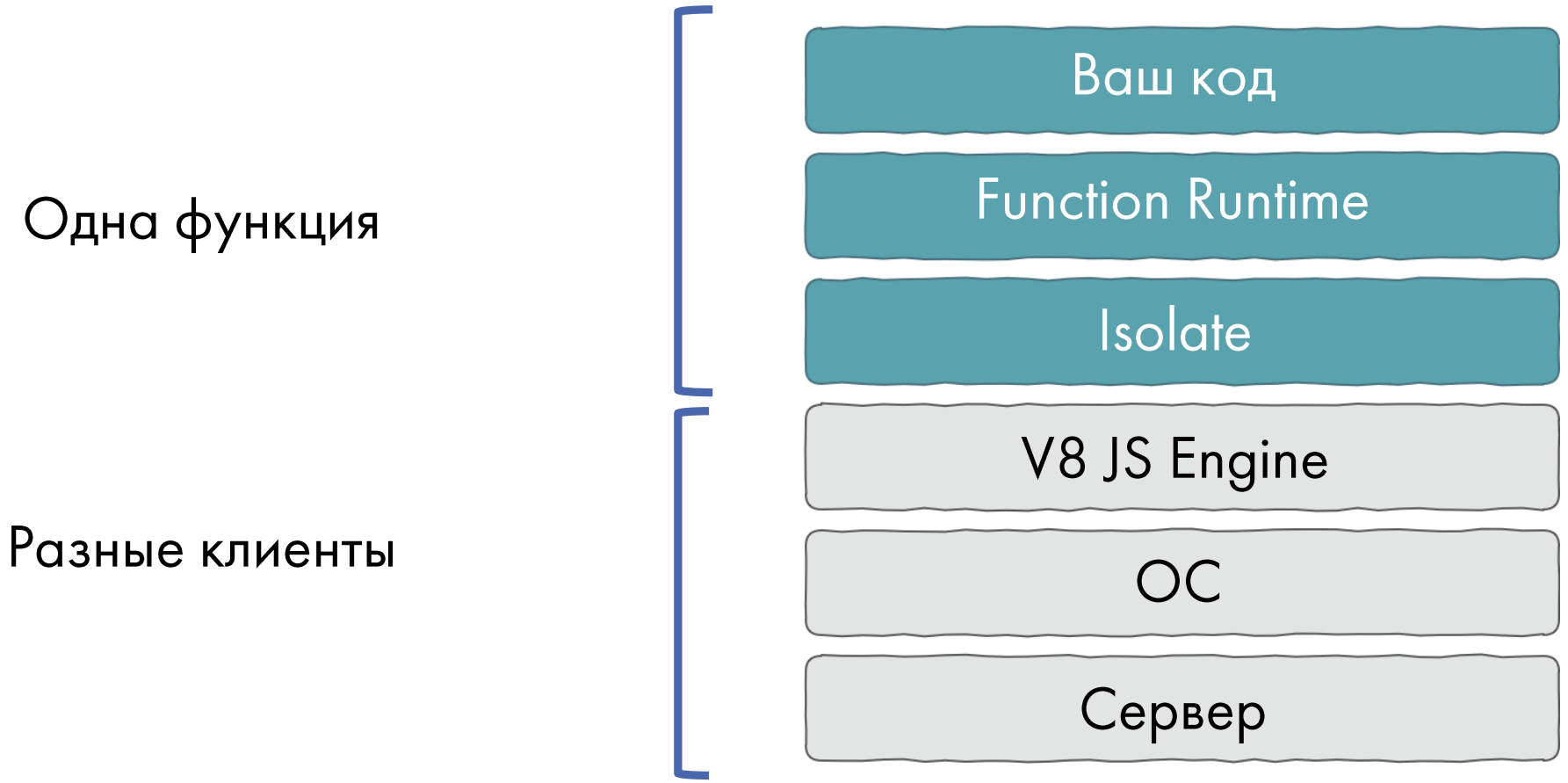
Google Cloud Functions

Один обработчик
(Container)

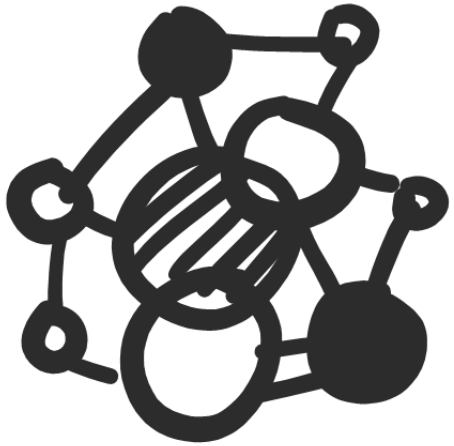
Разные клиенты
(Orchestrator)



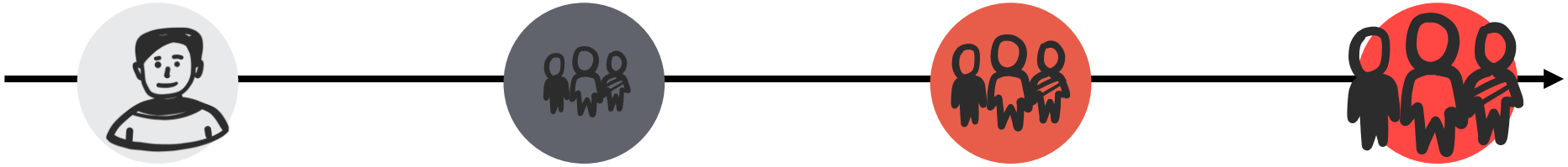
Cloudflare Workers



На практике



Идеальная масштабируемость: Пропускная способность

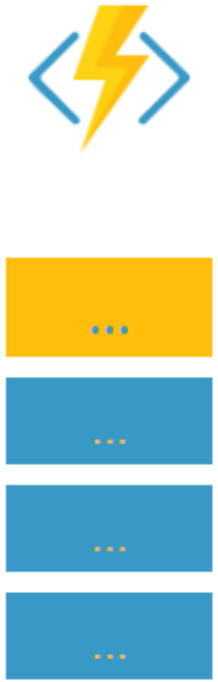


пользователи

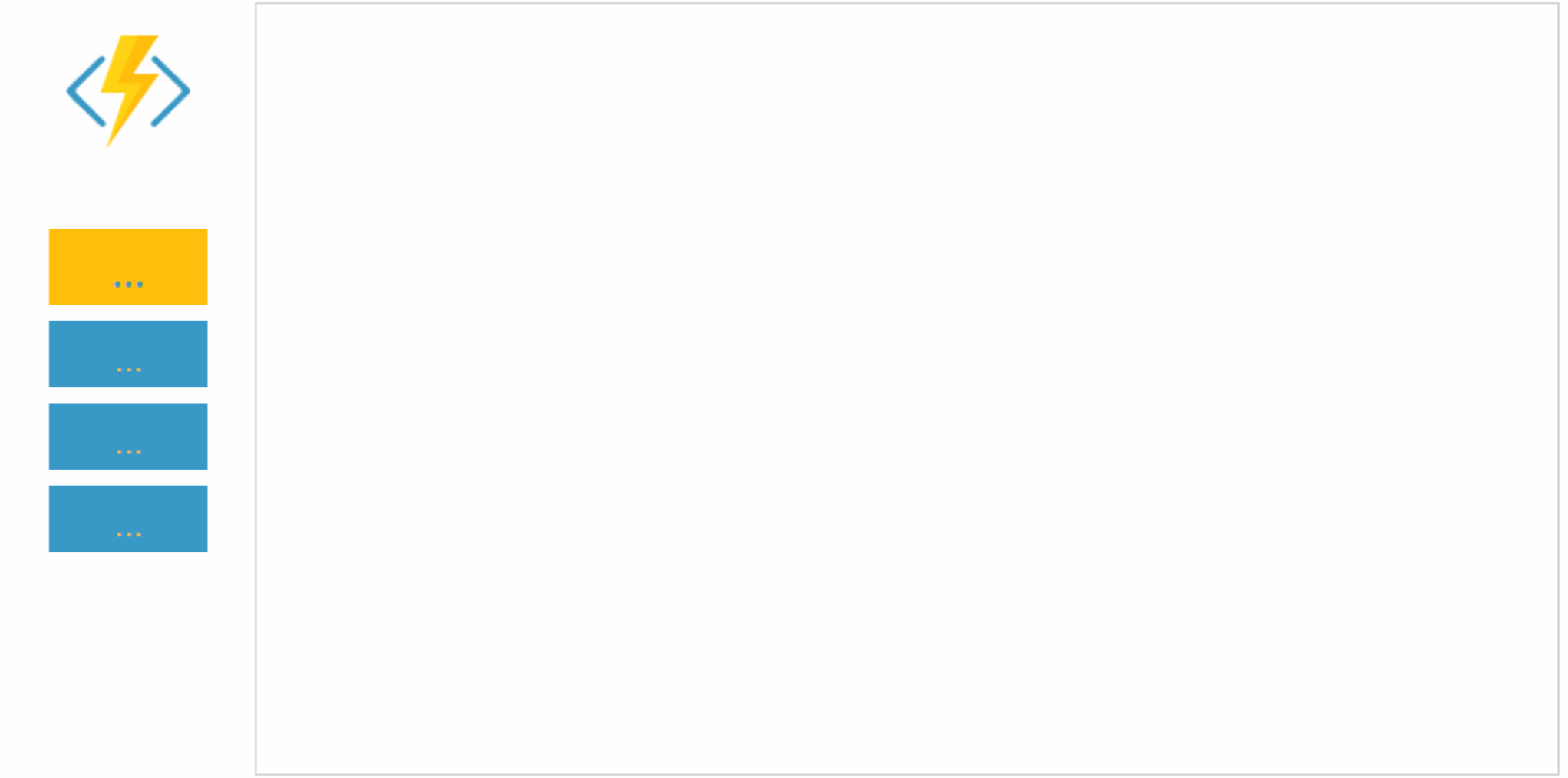
Идеальная масштабируемость: Задержка

Холодный Старт

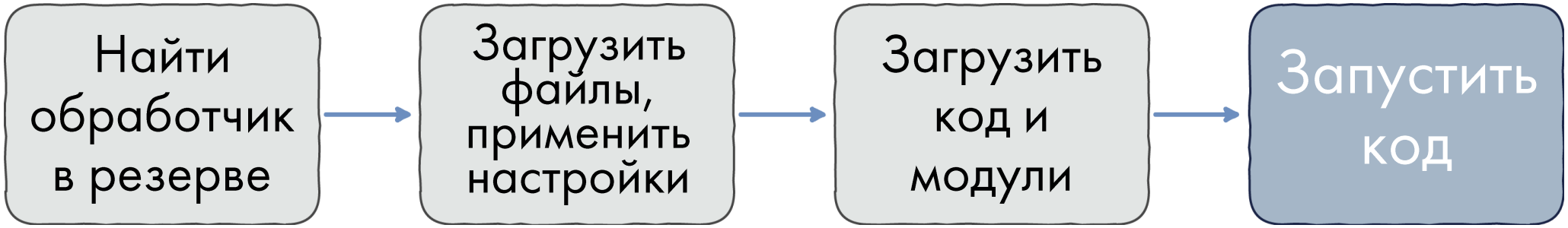
Пример: Загрузка карты



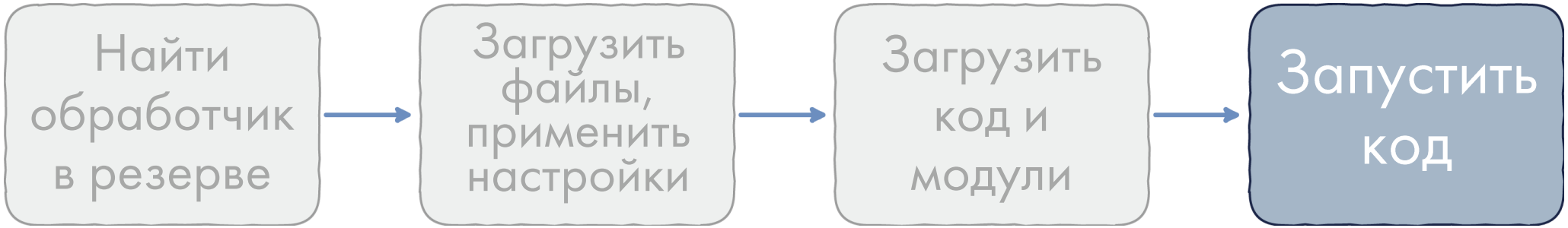
Пример: Загрузка карты



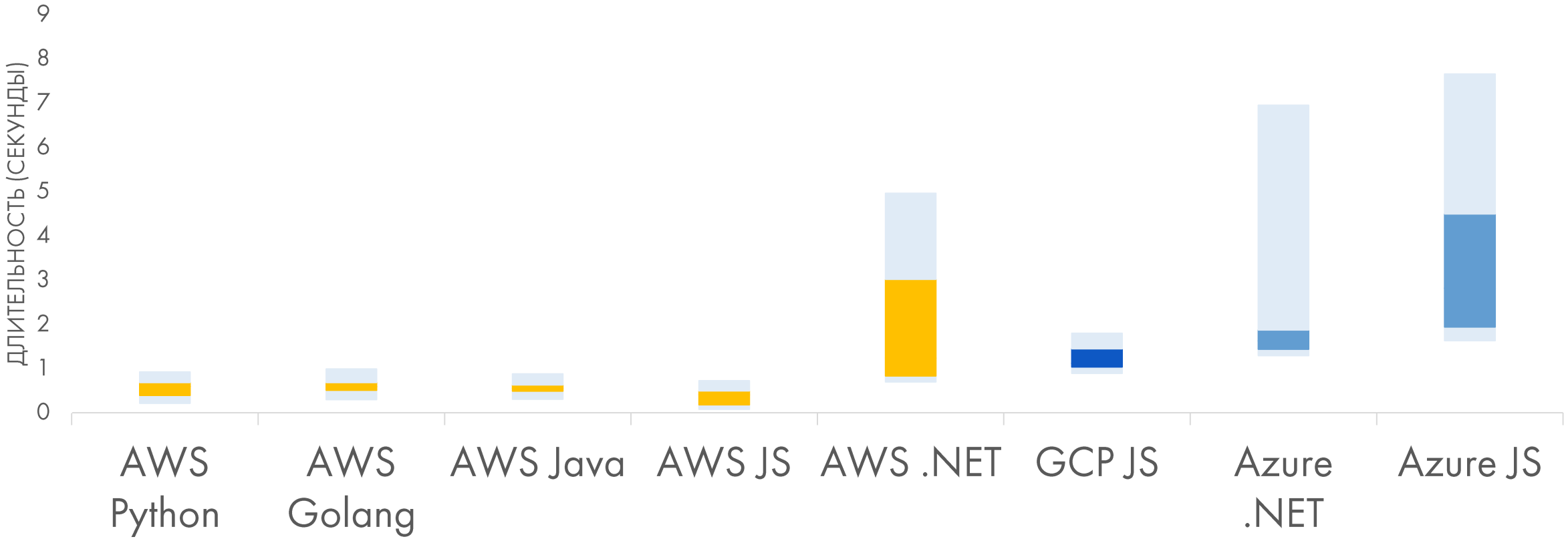
Холодный Старт



Горячий старт



Холодный старт: сравнение



Зависимости увеличивают время старта

Указаны размеры пакета установки - zip архива

Период без запросов, приводящий к Холодному старту

25-60 минут

20 минут

Обычно
5+ часов

Pre-JITTed runtime

CrossGen

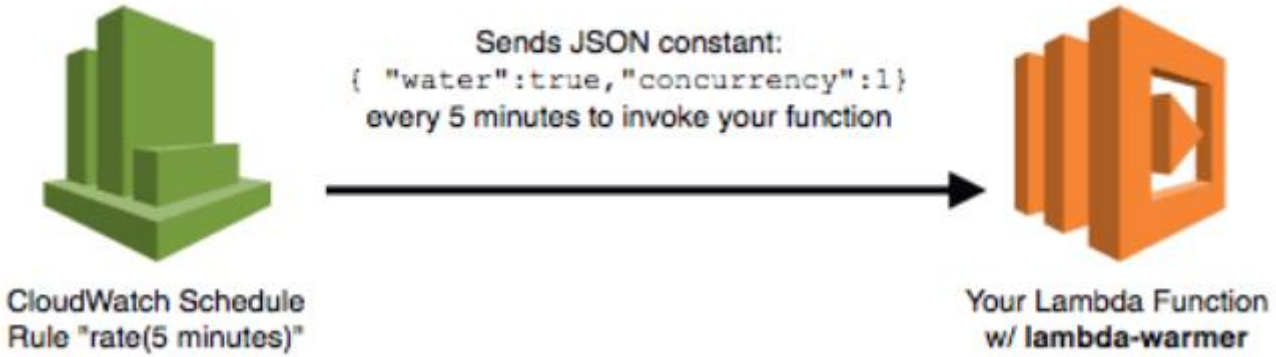
Lambda Layers

Binding пакеты

Борьба с холодным стартом: JIT оптимизации

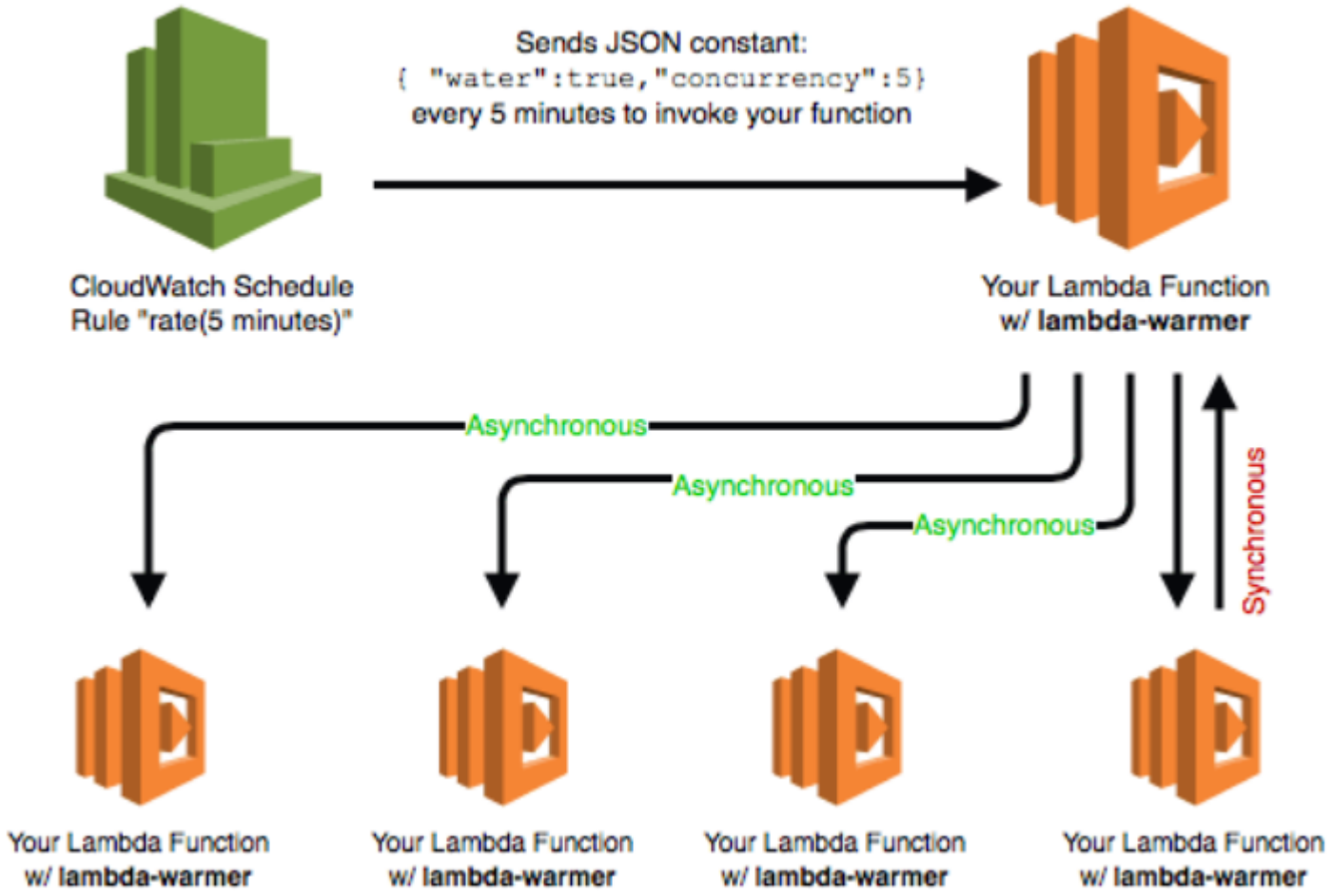
Борьба с холодным стартом: Azure

```
[FunctionName("Warmer")]
public static void WarmUp(
    [TimerTrigger("0 */15 * * * *")]
    TimerInfo timer)
{
    // Пусто
}
```



AWS: Прогрев одного обработчика

https://github.com/jeremydaly/lambda-warmer



AWS: Прогрев нескольких обработчиков

https://www.jeremydaly.com/lambda-warmer-optimize-aws-lambda-function-cold-starts/

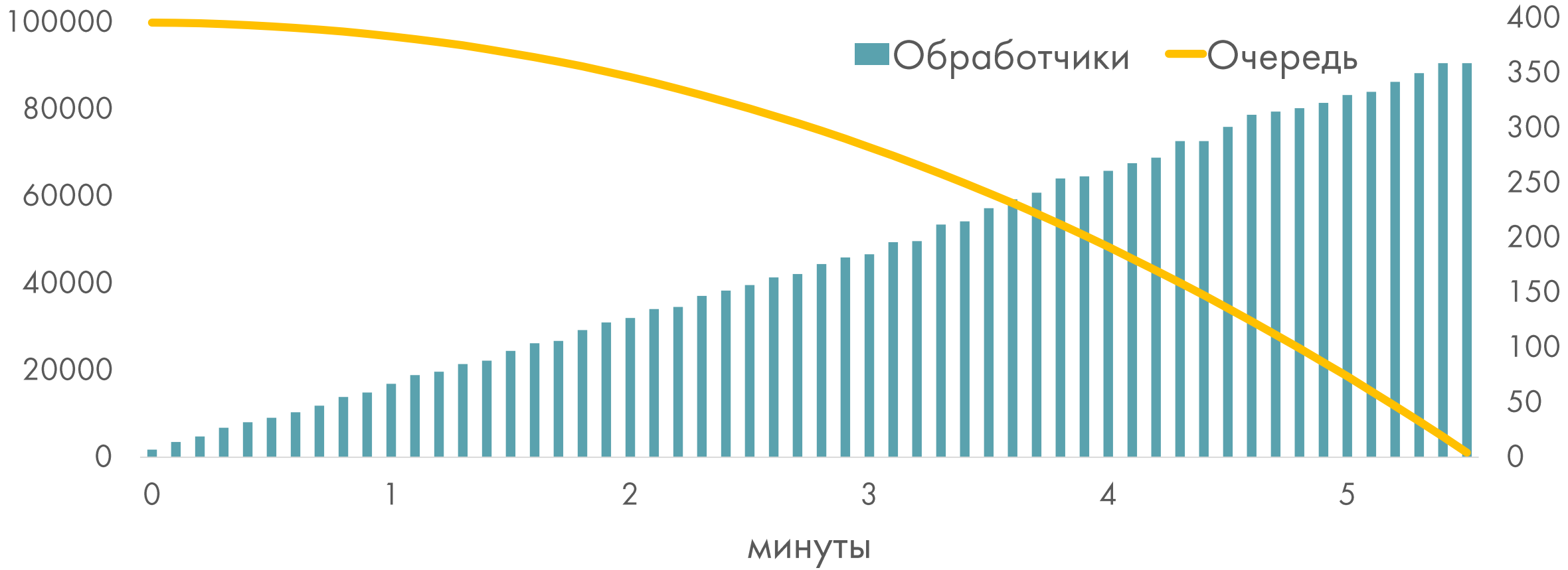
Serverless ETL:
Асинхронная обработка данных

Типичный сценарий

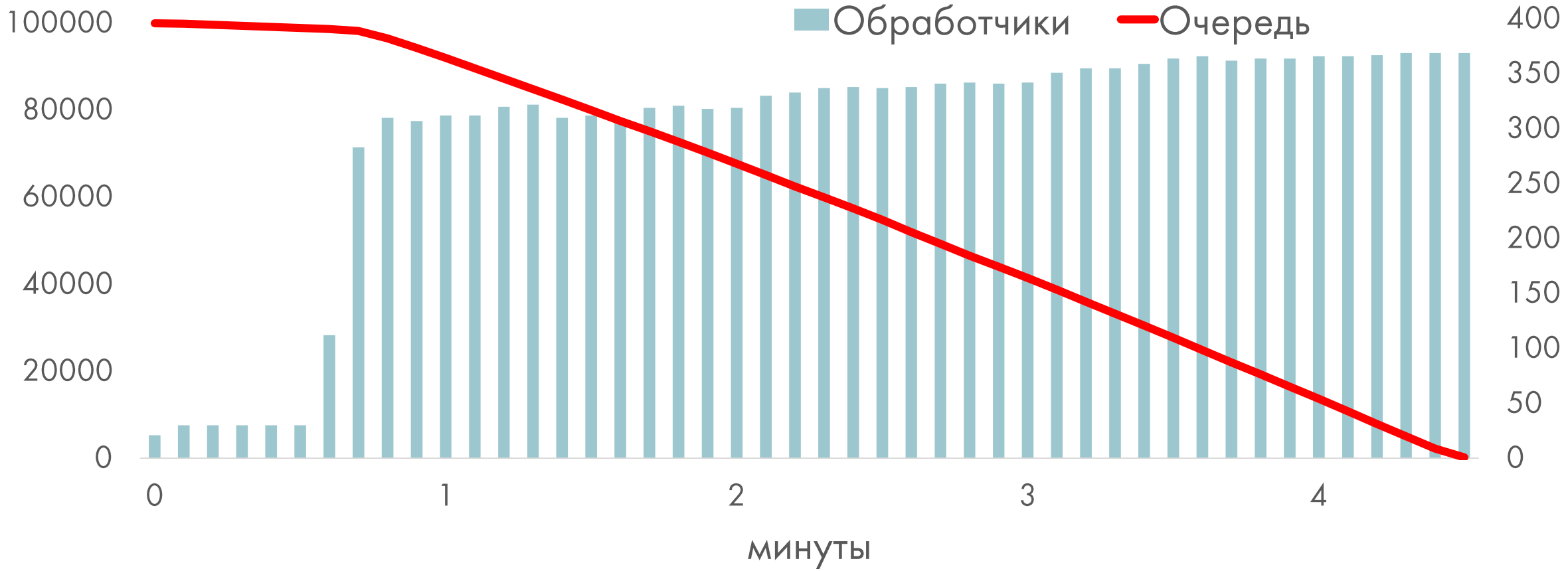
Эксперимент с очередями

Сценарий	Тест 1	Тест 2
100 000 сообщений добавляются в очередь. Как быстро они будут обработаны?	Функция: **sleep(500 мс).** Эмулирует ожидание сети (I/O Bound).	Функция: **BCrypt hash.** CPU Bound.

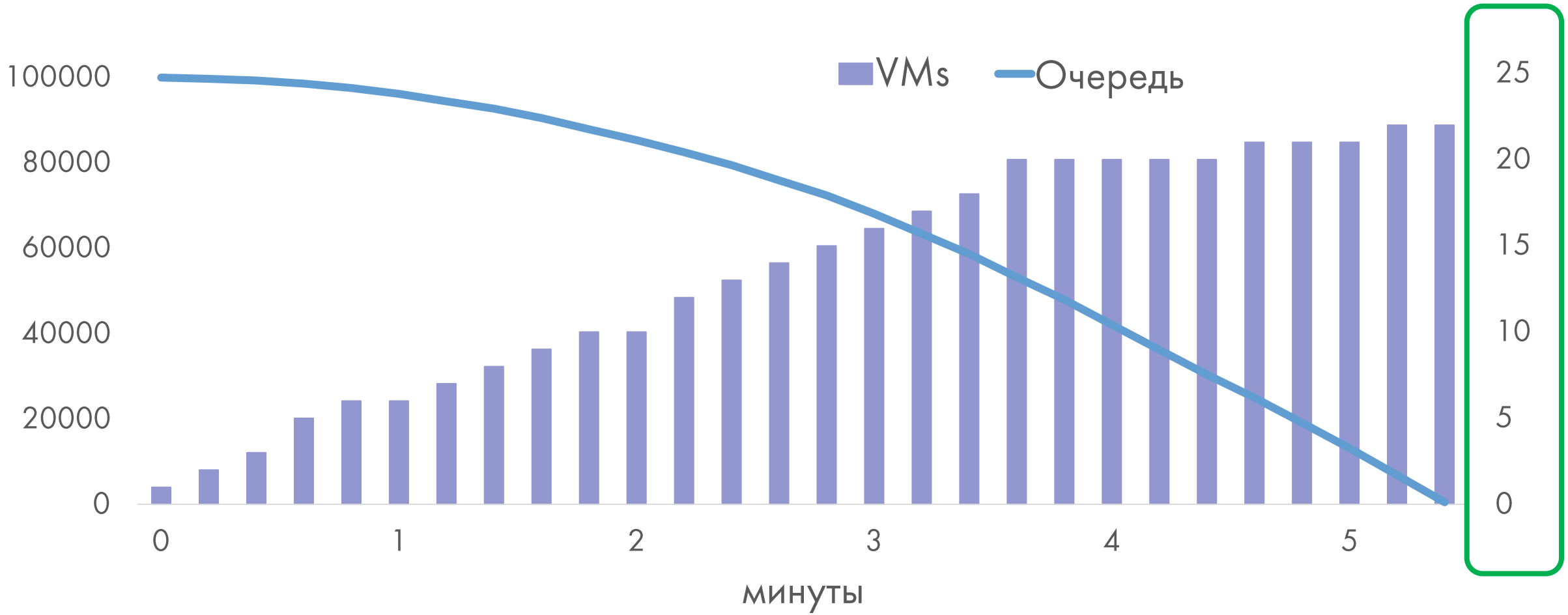
AWS Lambda и 100 тыс. SQS сообщений



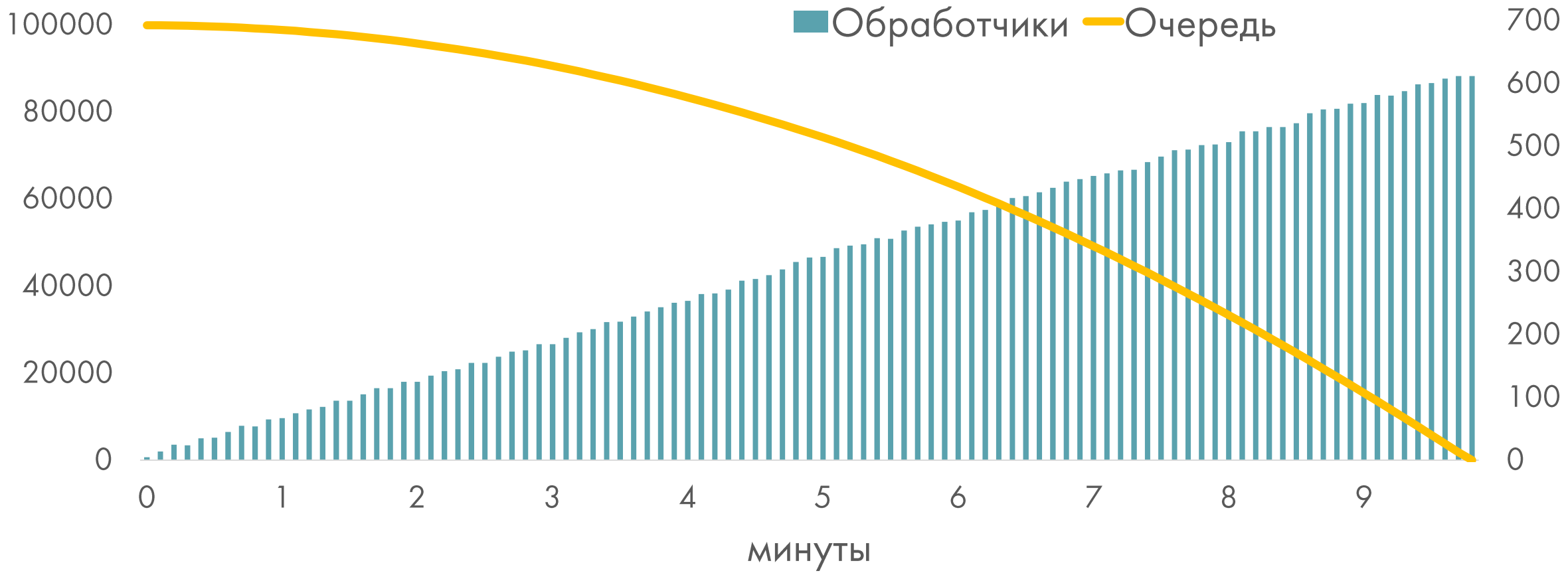
GCF и 100 тыс. Pub/Sub сообщений



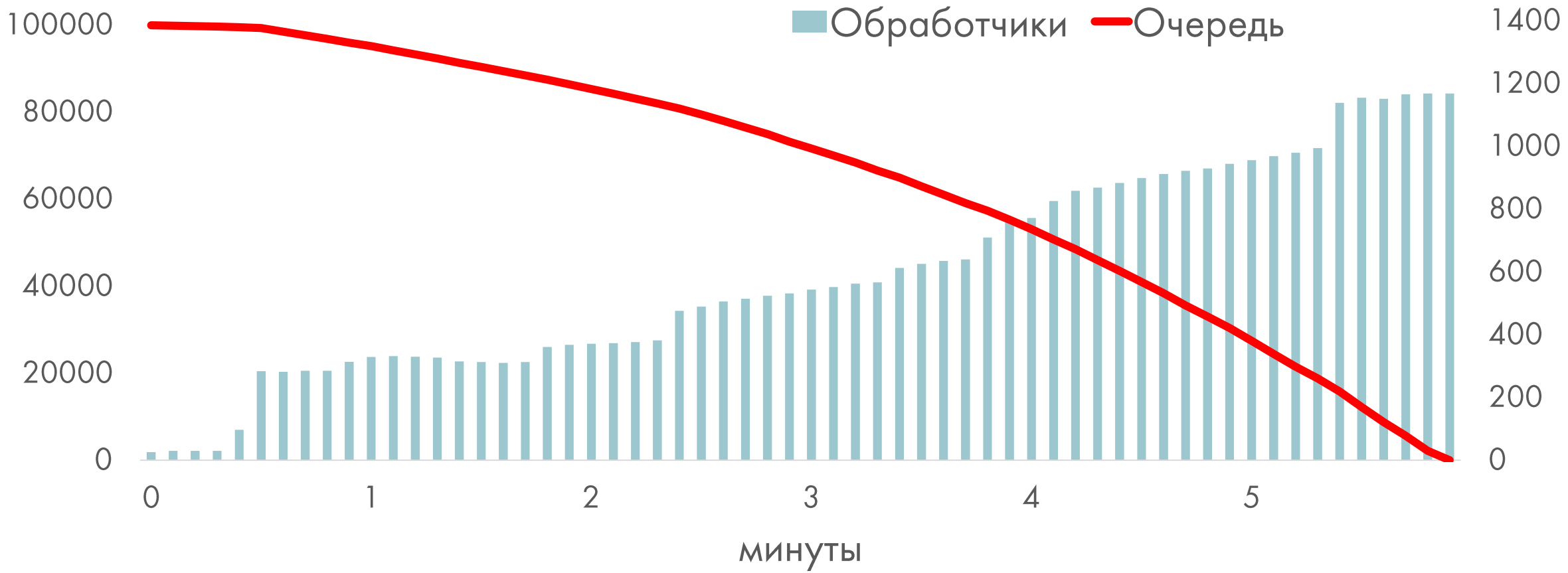
Azure Functions и 100 тыс. Storage Queue сообщений



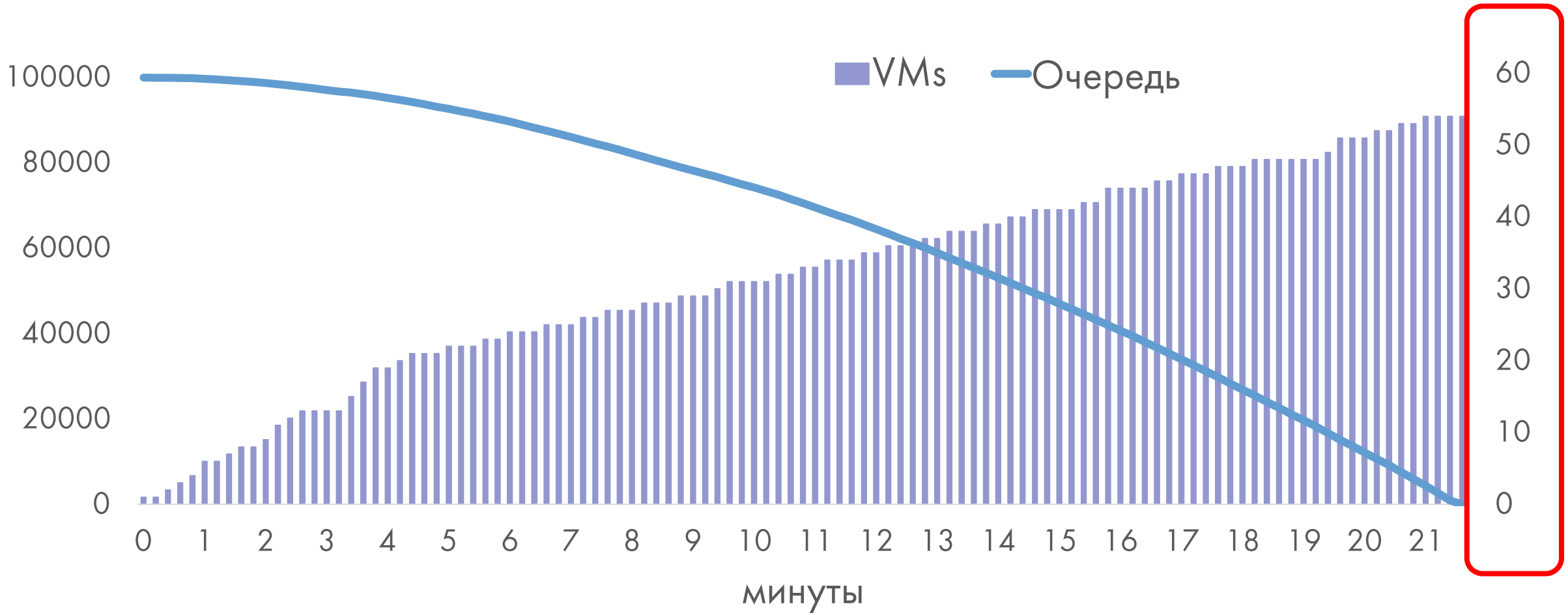
Требовательная к CPU нагрузка на AWS Lambda



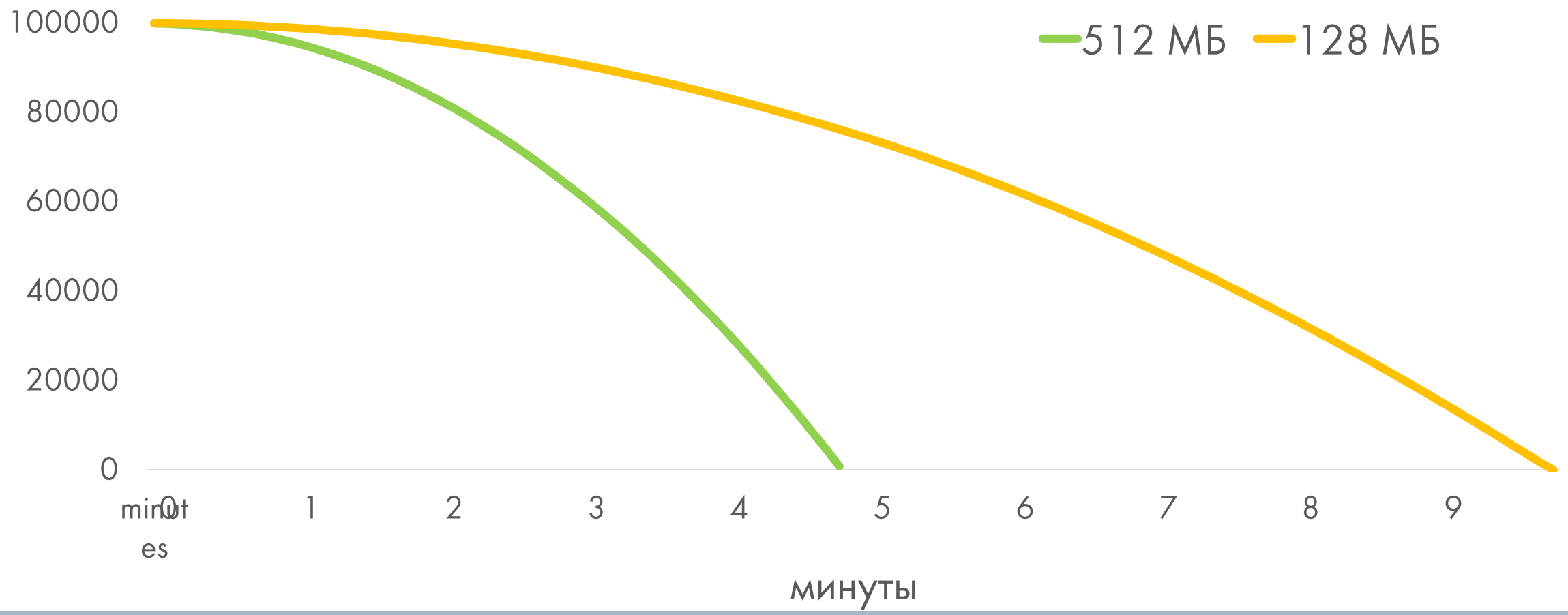
Требовательная к CPU нагрузка на GCF



Требовательная к CPU нагрузка на Azure Functions

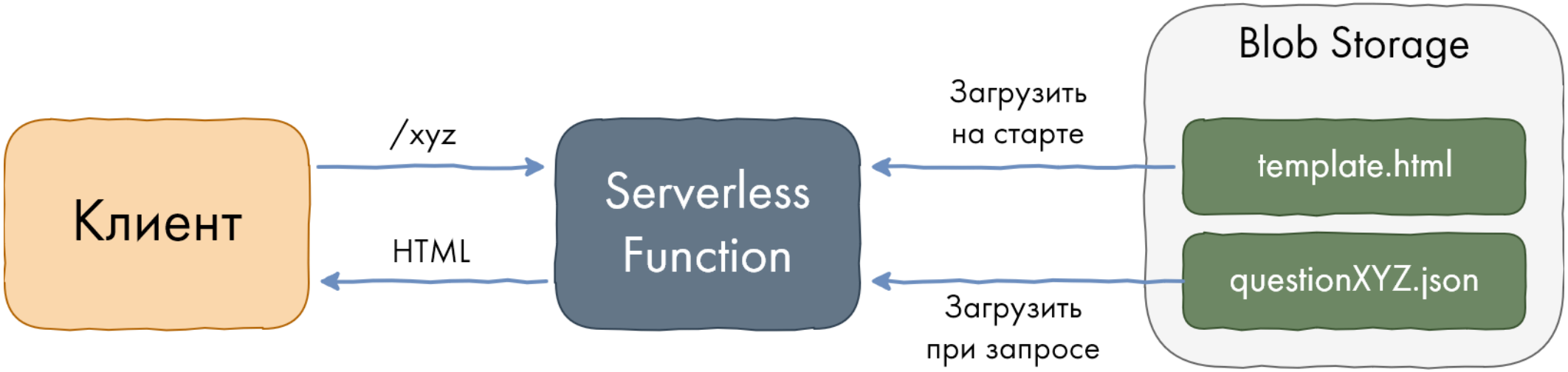


AWS скорость обработки зависит от резевра RAM

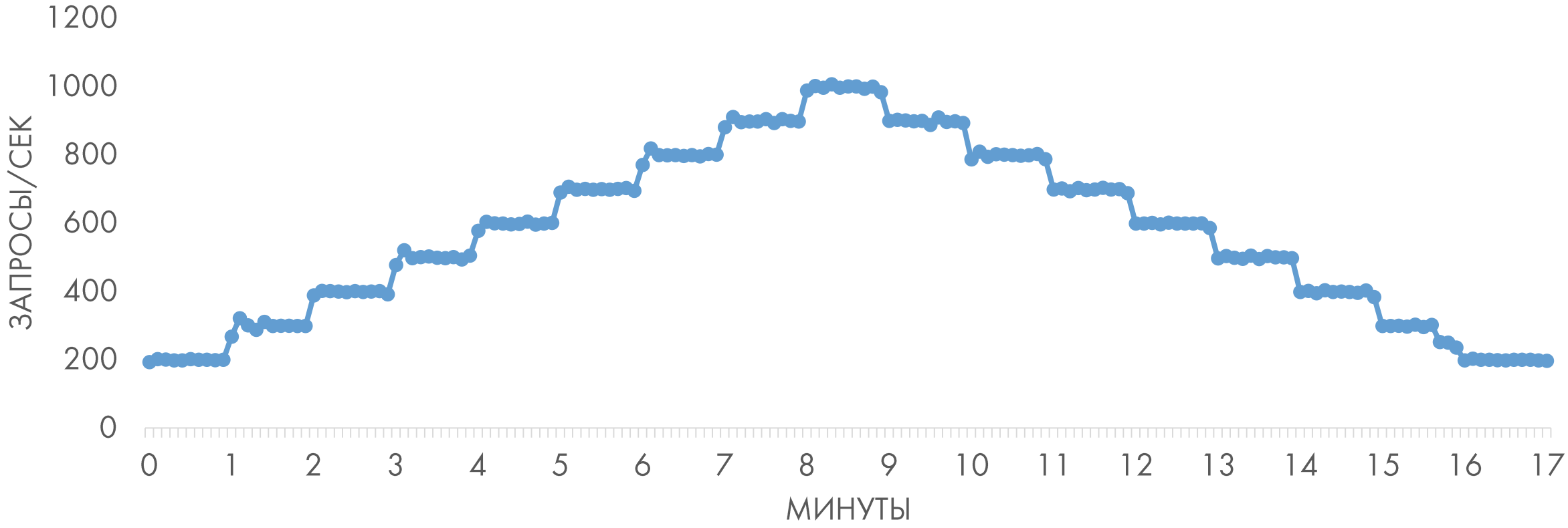


Serverless HTTP:
Обработка трафика а-ля StackOverflow

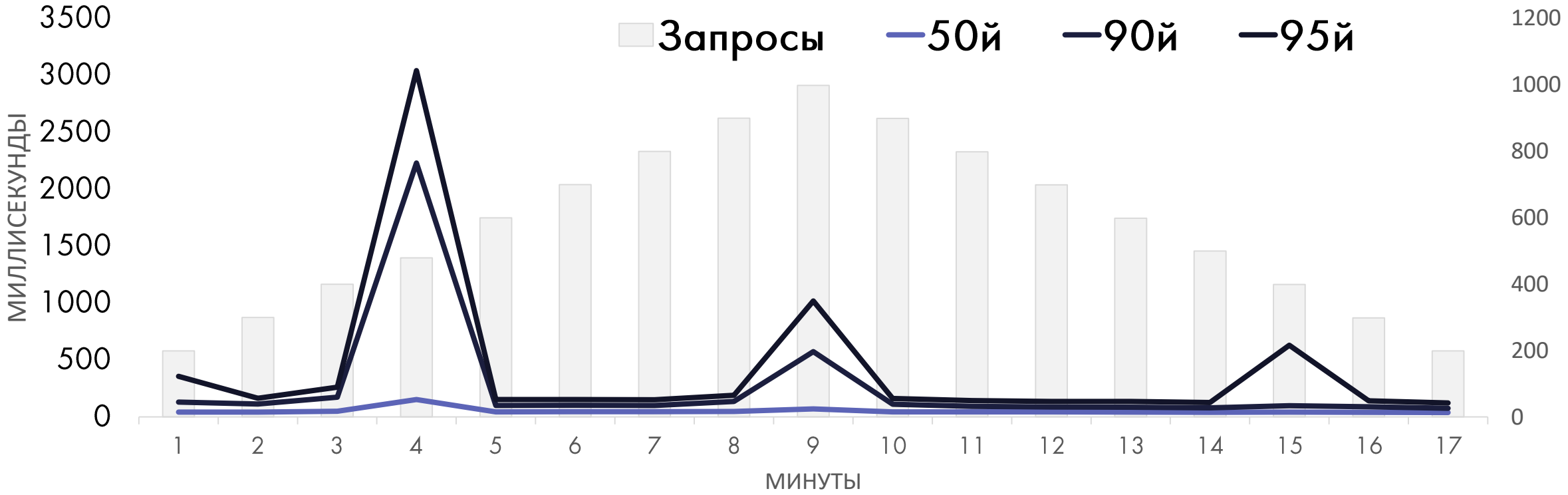
Подражаем StackOverflow



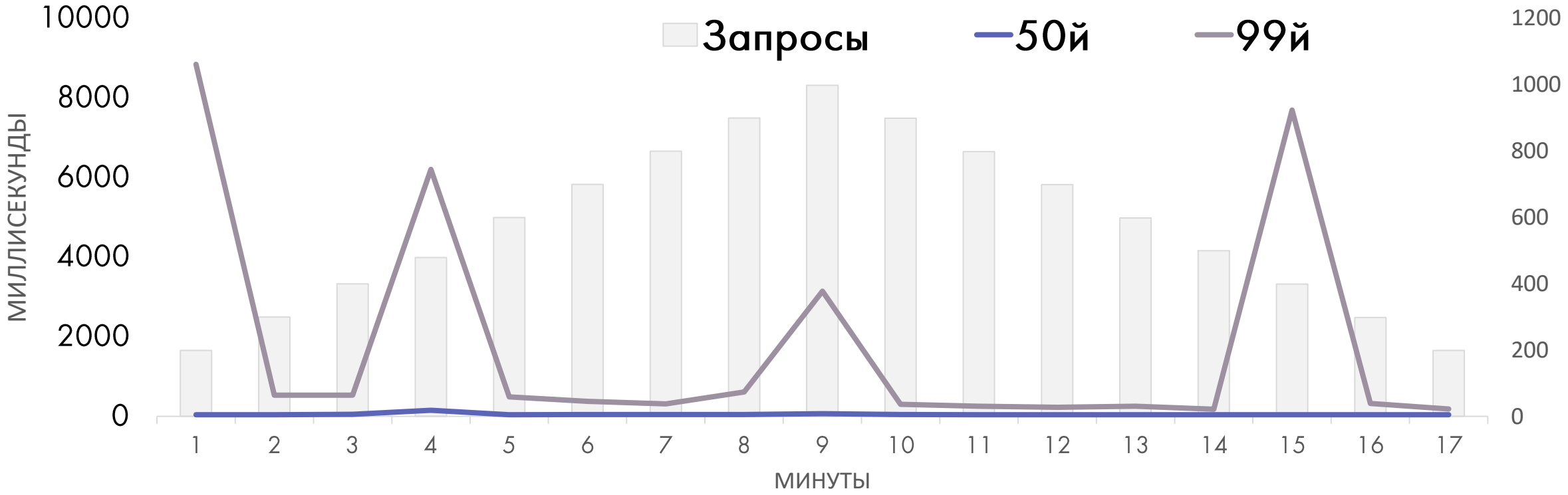
Запросы/сек во время теста



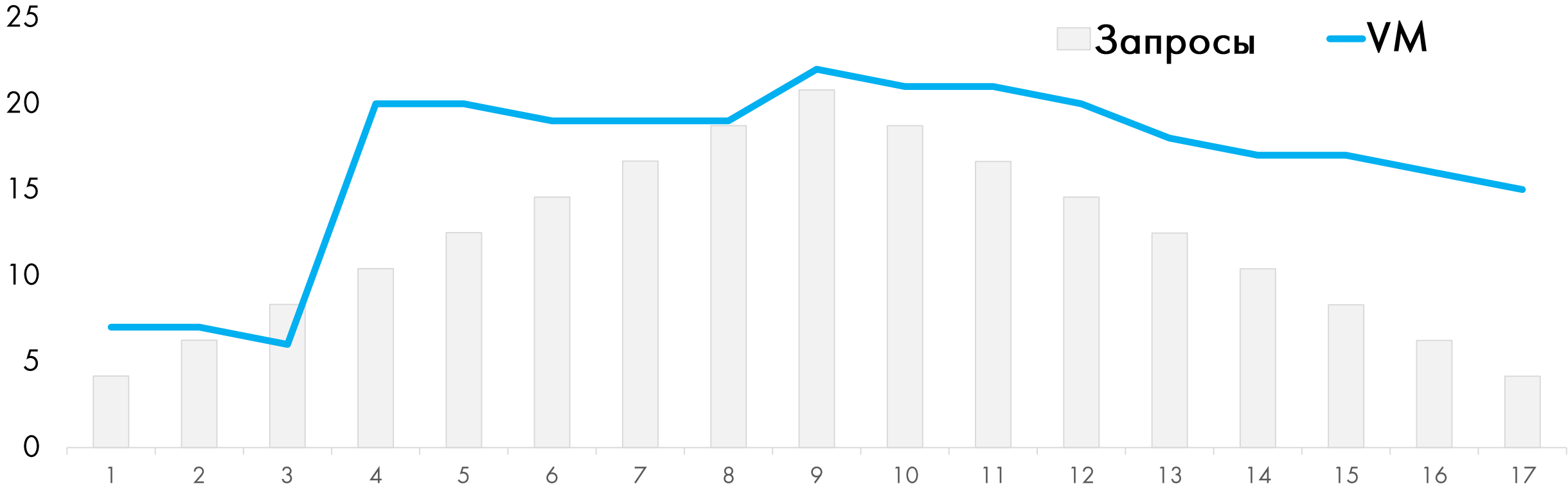
Azure Functions: процентили



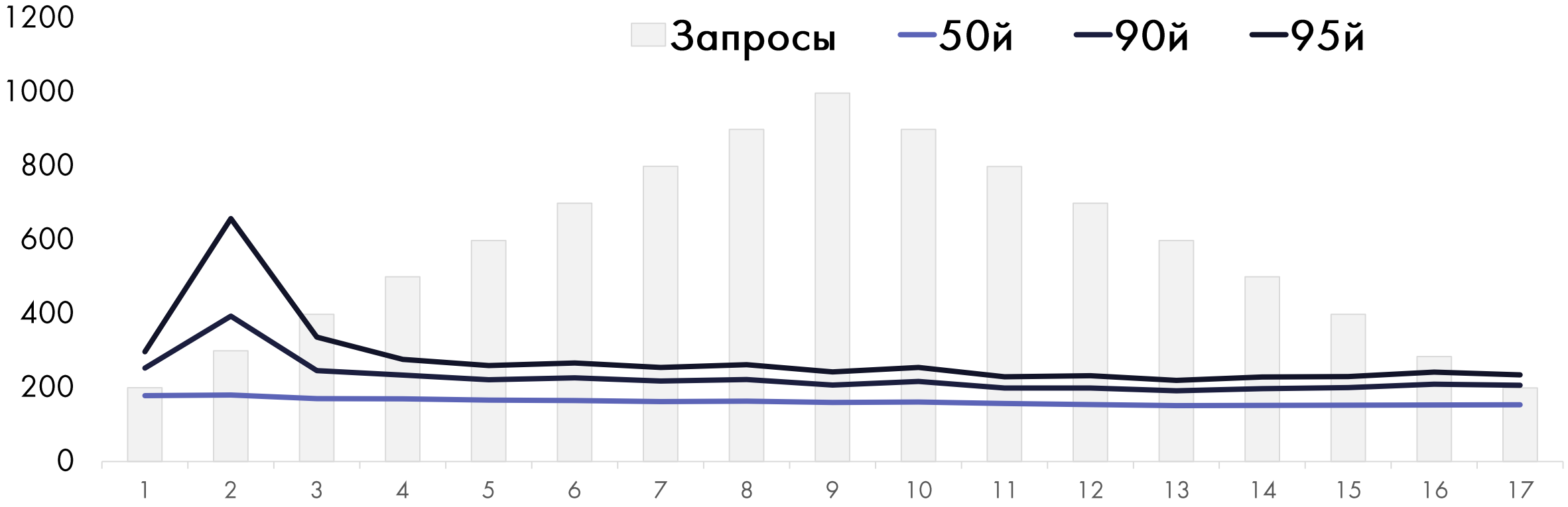
Azure Functions: процентили



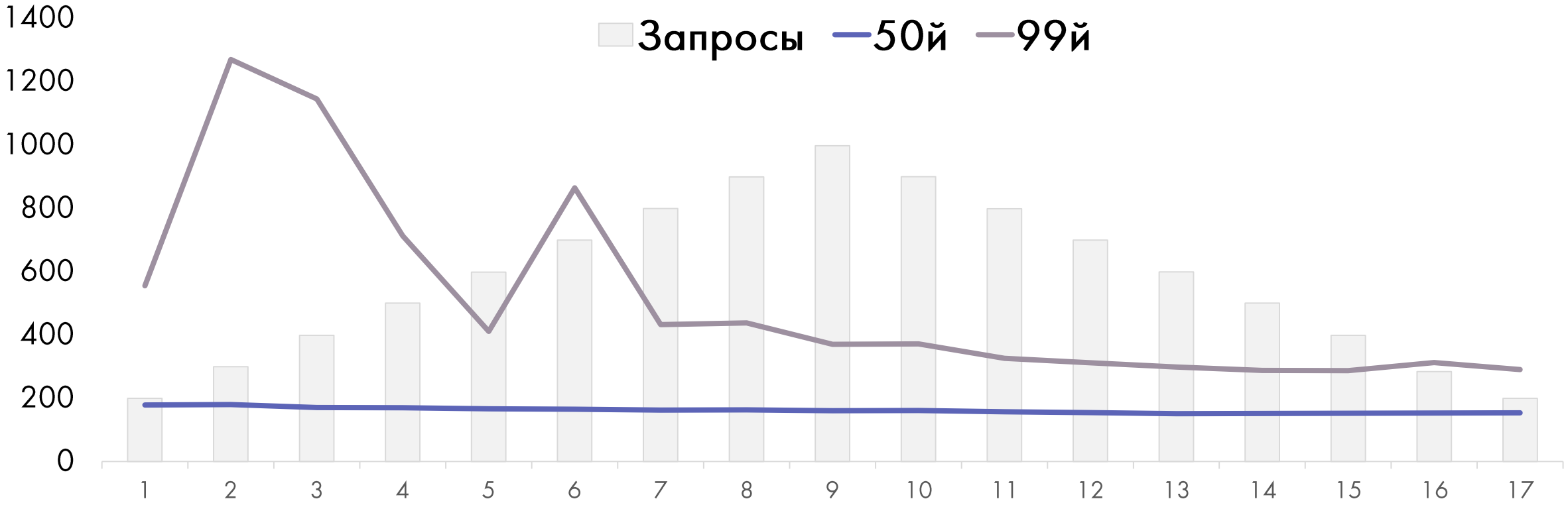
Azure Functions: кол-во VM



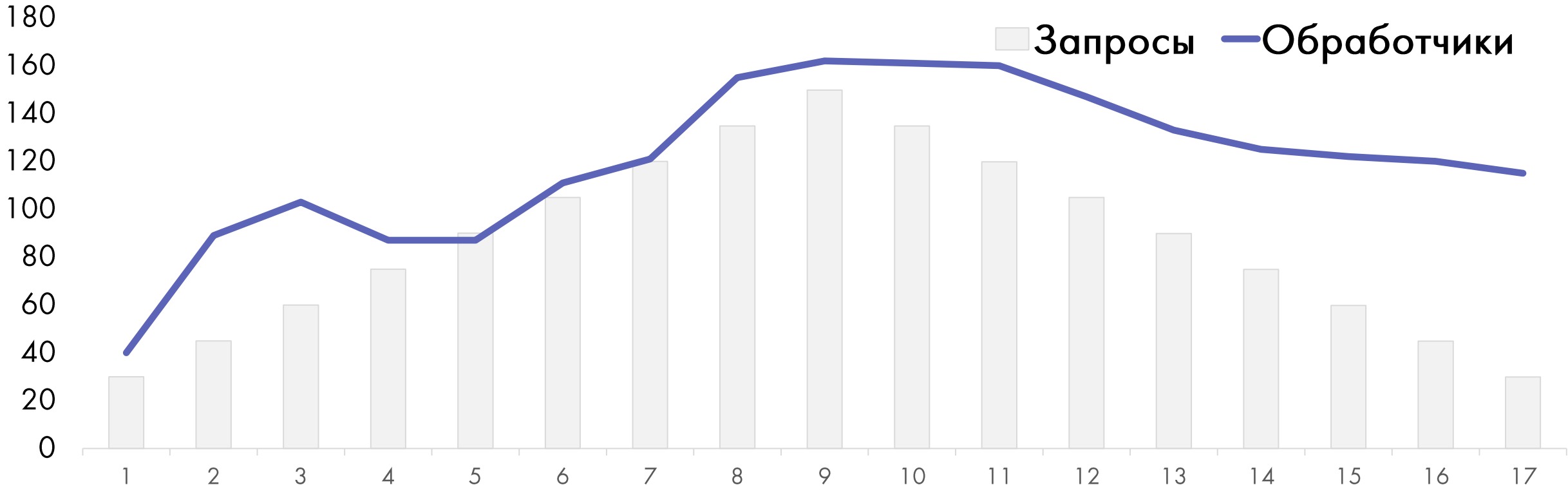
Google Cloud Functions: процентили



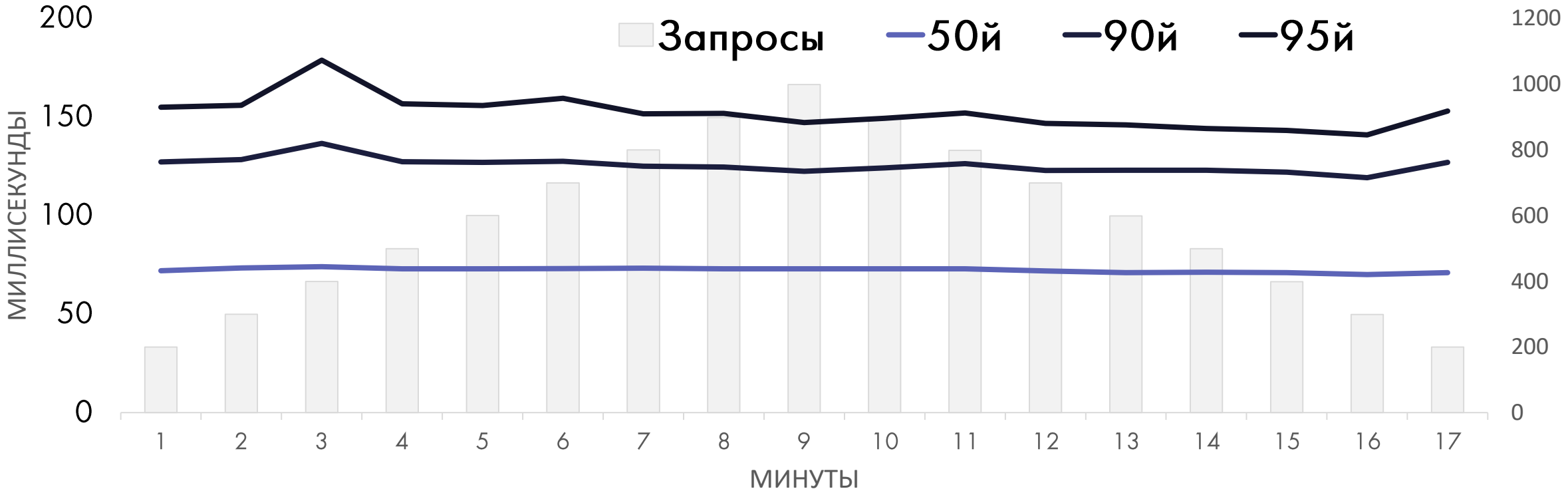
Google Cloud Functions: процентили



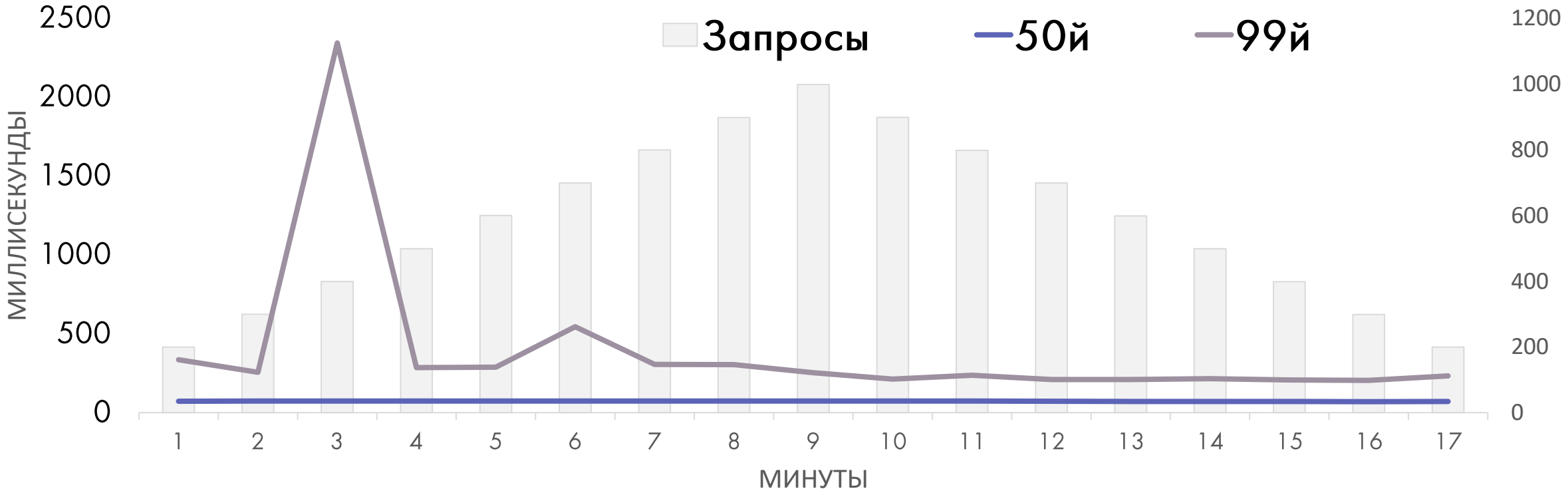
Google Cloud Functions: обработчики



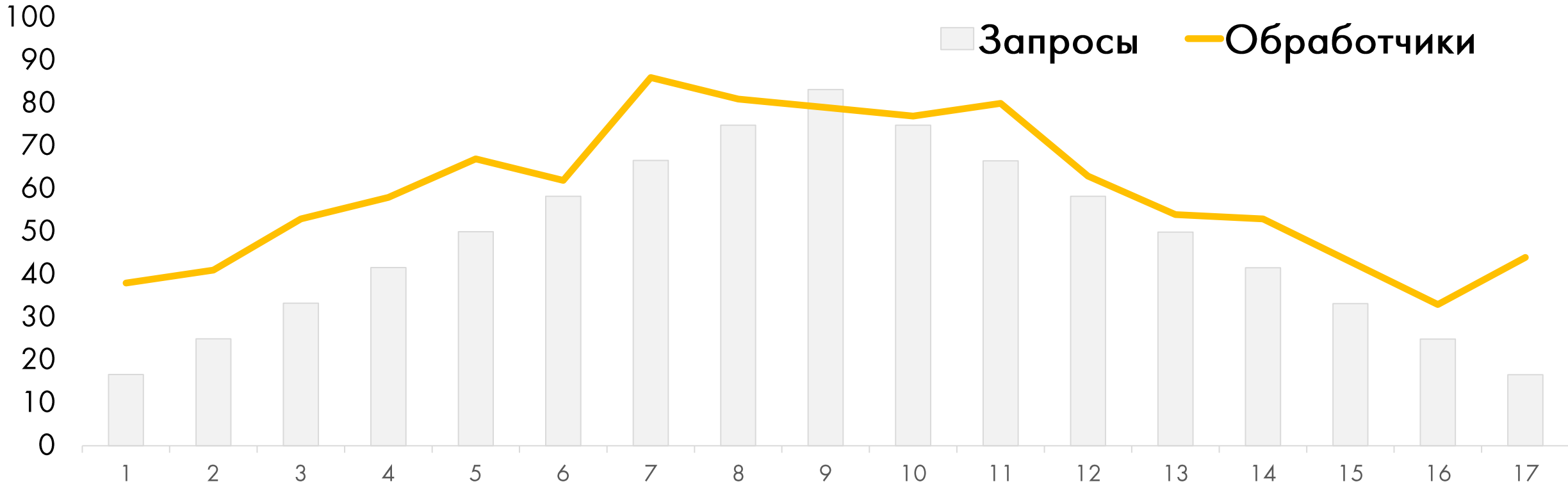
AWS Lambda: процентили



AWS Lambda: процентили



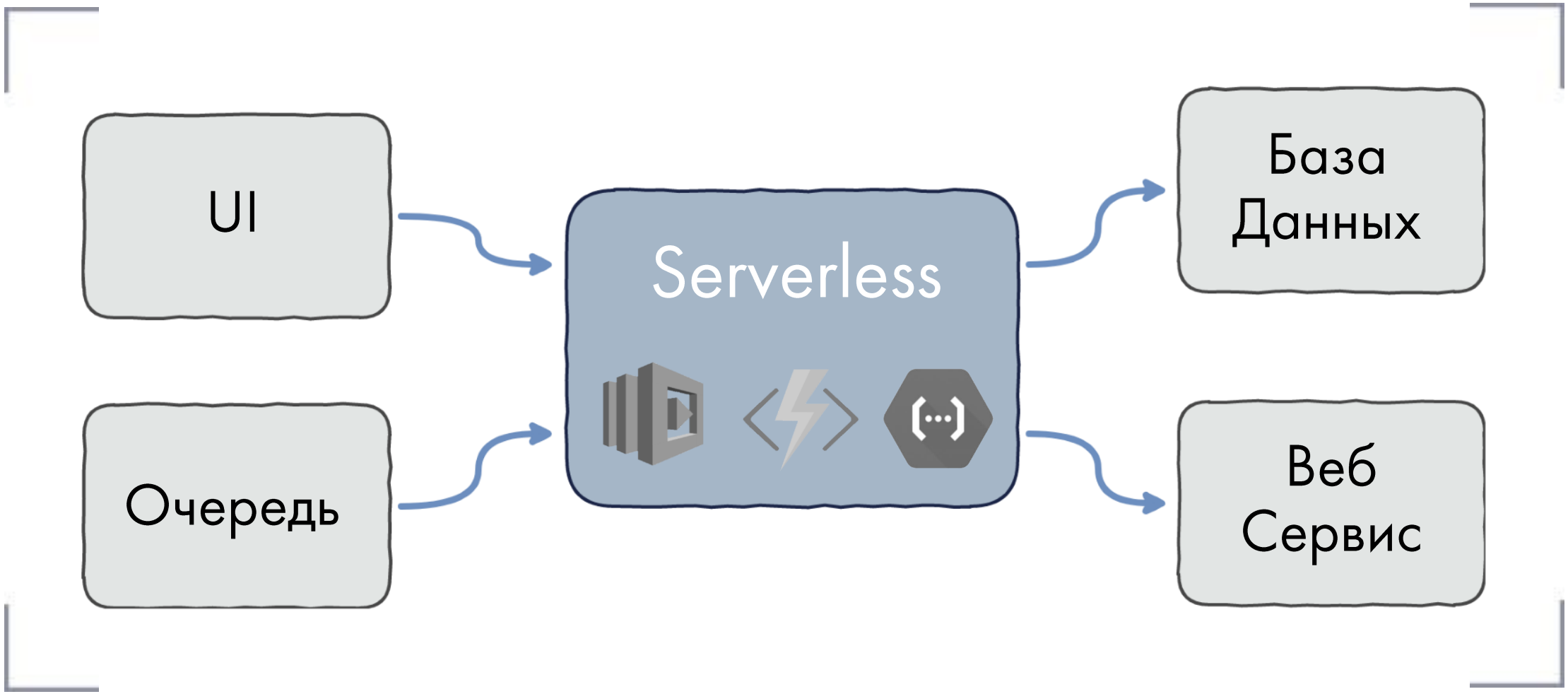
AWS Lambda: обработчики



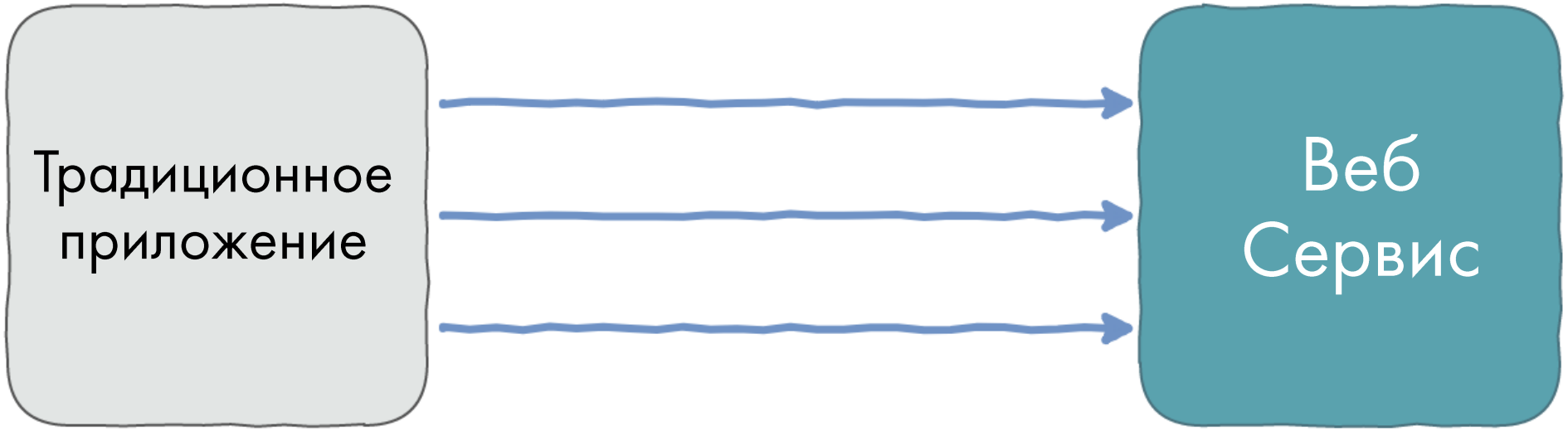
Система:

взаимодействие с не-serverless компонентами

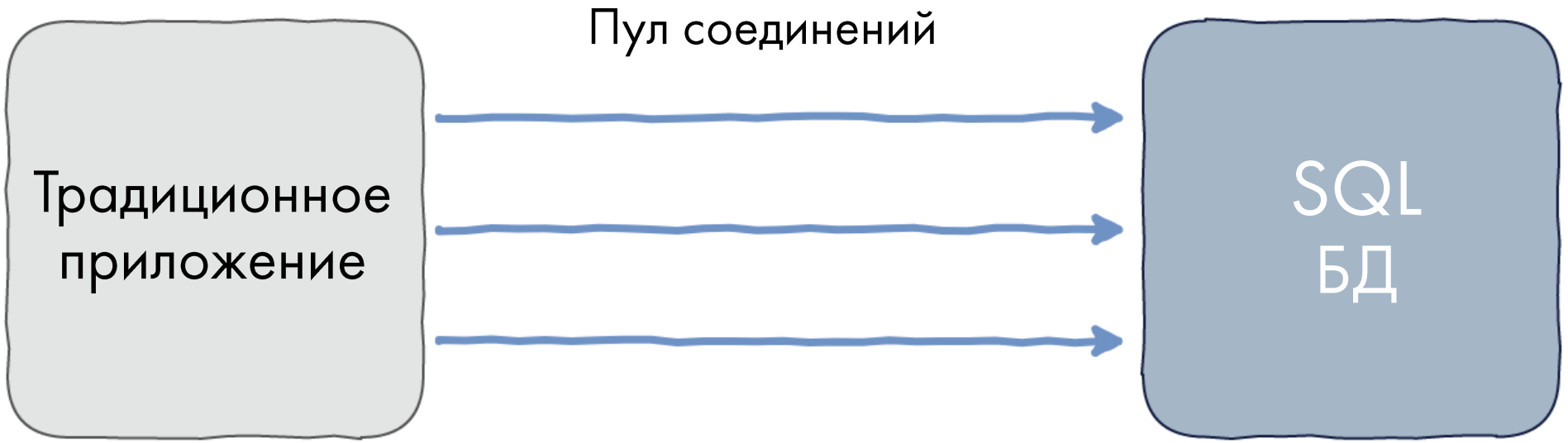
Типичное приложение



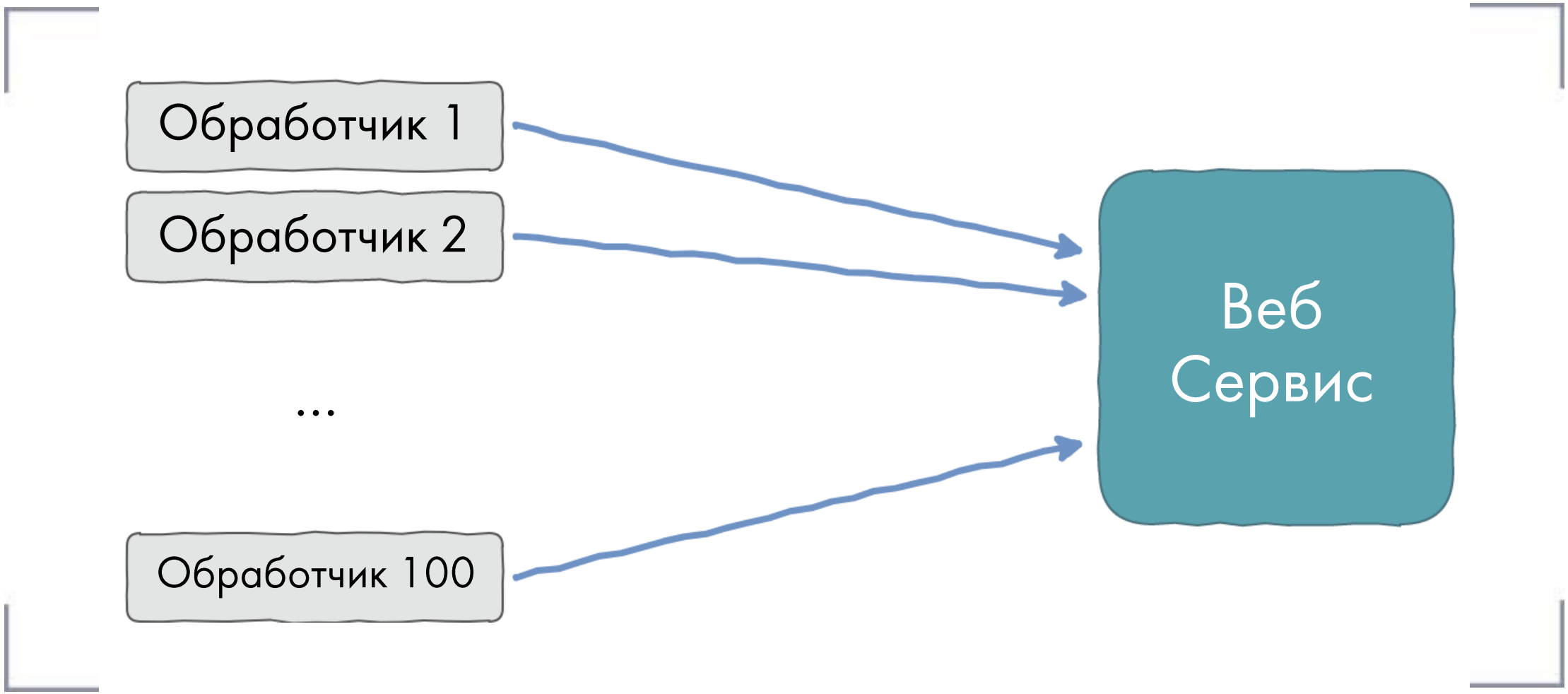
Веб Сервисы



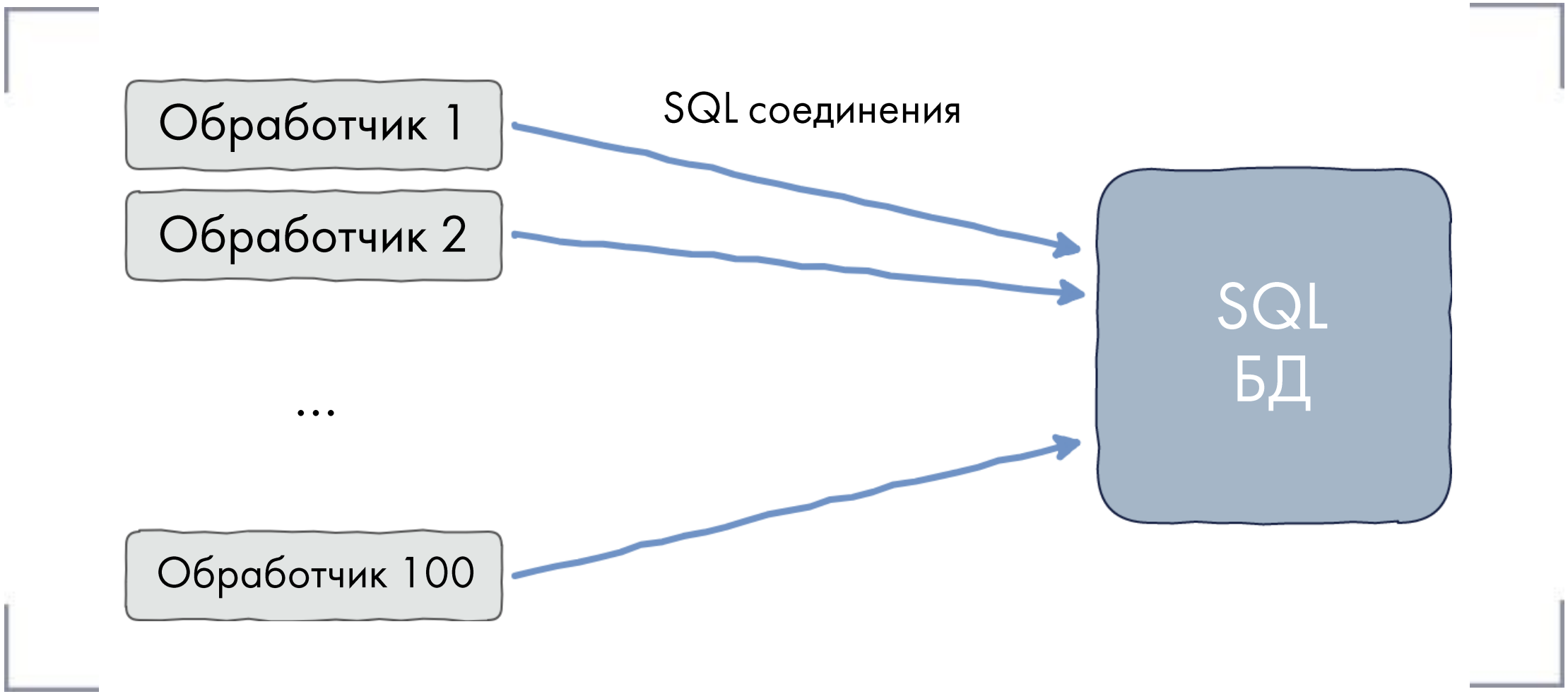
Базы Данных

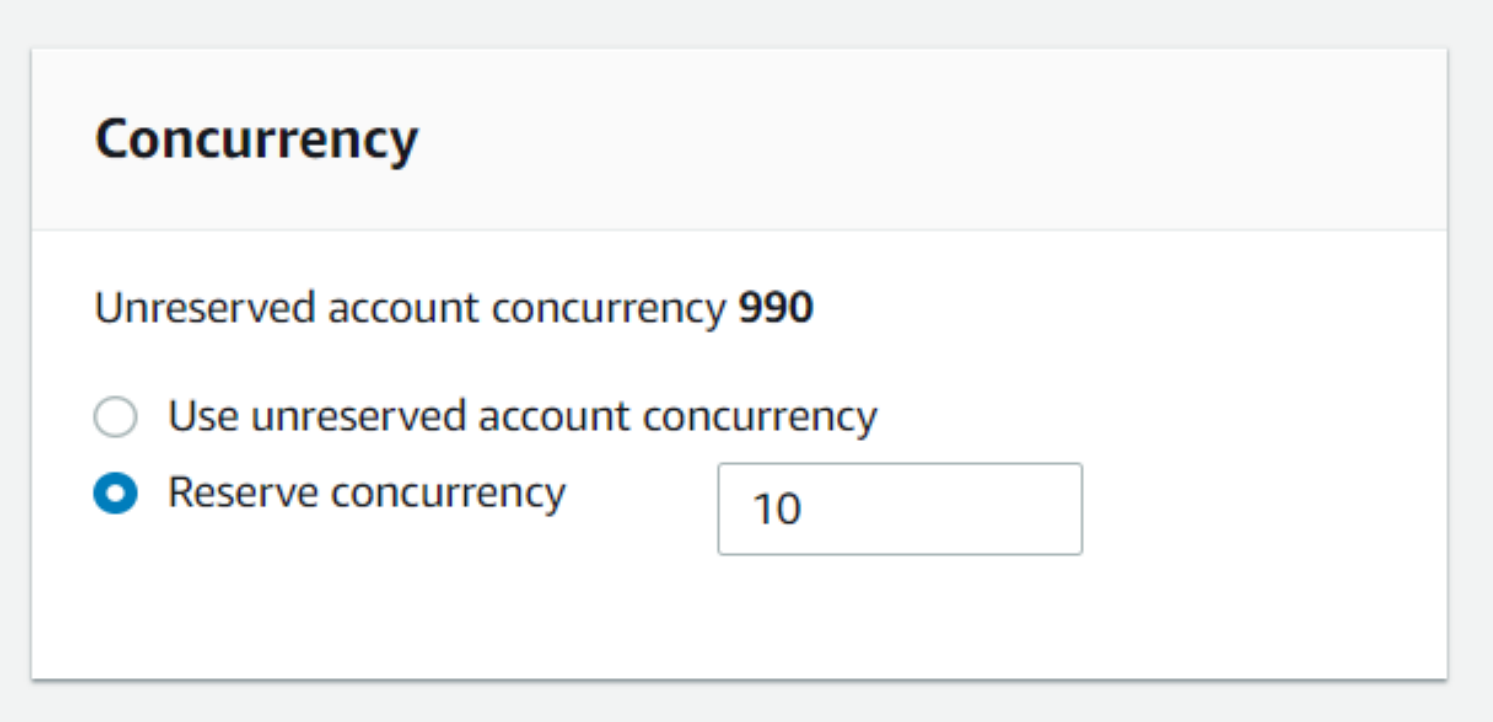


Serverless масштабирование



Serverless масштабирование





AWS: Максимальное количество параллельных выполнений

WEBSITE_MAX_DYNAMIC_APPLICATION_SCALE_OUT

The maximum number of instances that the function app can scale out to. Default is no limit.

> **Note**
>
> This setting is a preview feature - and only reliable if set to a value <= 5

Key	Sample value
WEBSITE_MAX_DYNAMIC_APPLICATION_SCALE_OUT	5

Azure: Максимальное количество VM

Может не работать для значений > 5

Повторное использование HTTP соединений

```
private static HttpClient Client = new HttpClient();

[FunctionName("MyFunc")]
public static async Task Run([QueueTrigger("q")] string message)
{
  var response = await Client.GetAsync("https://mikhail.io");
  // ... the rest goes here
}
```

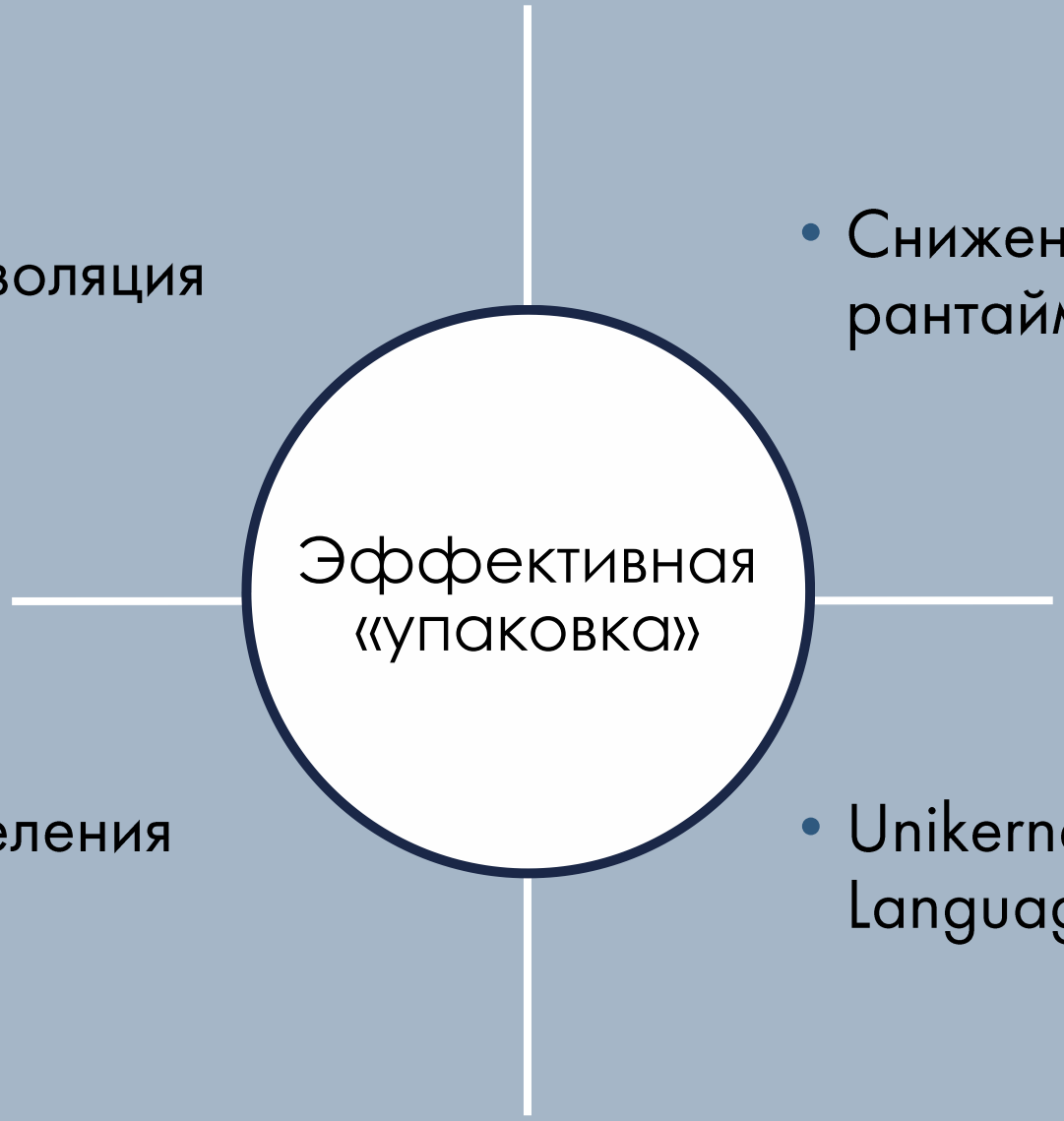
Какие задачи предстоит решить?

Заглядывая в будущее

- Поддержка приложений, чувствительных к задержкам
- Создать обработчик до того, как он будет необходим

Холодные старты

- Пример: Cloudflare
- Пример: Binaris



- Безопасная изоляция
- Снижение «веса» рантайма
- ML для определения компоновки
- Unikernels, library OSes, Language VMs?

- Вычисления рядом с данными
- «Stateful» приложения
- Высокопроизводительное хранилище данных
- Специализированное оборудование (GPU/TPU)


Thank you!
@MikhailShilkov
https://mikhail.io
© Mikhail Shilkov