Микросервисы: масштабирование

Михаил Триполитов
Банк Открытие, архитектор решений

План занятия

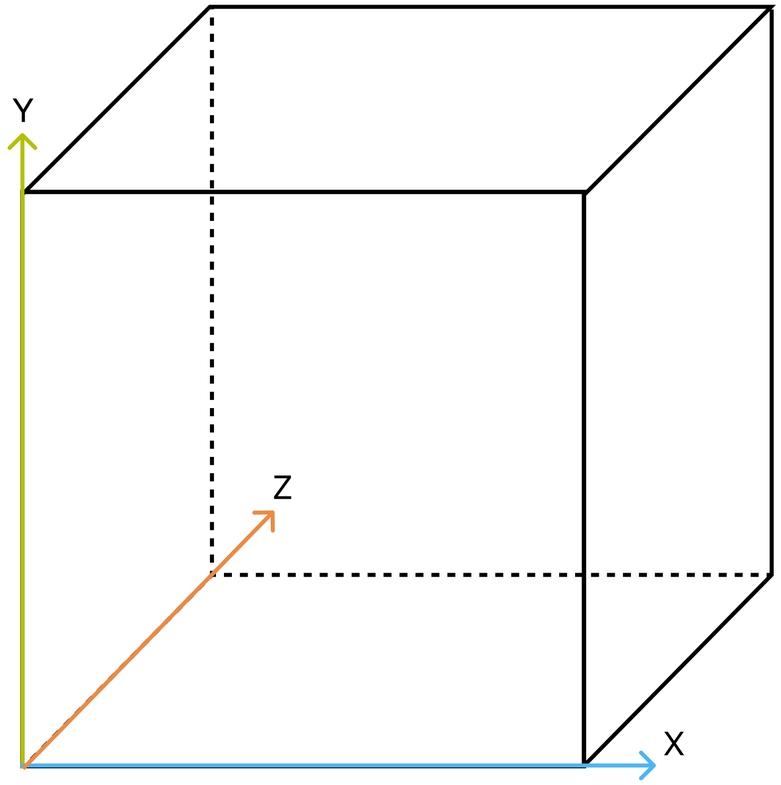
1. Хрупкость системы
2. Балансировка
3. Кеширование
4. Автомасштабирование
5. Service Discovery & Service Mesh

Scale Cube

Ось X — горизонтальное масштабирование

Ось Y — масштабирование разделением приложения

Ось Z — масштабирование через разделение данных



Хрупкость системы

Михаил Триполитов
Банк Открытие, архитектор решений

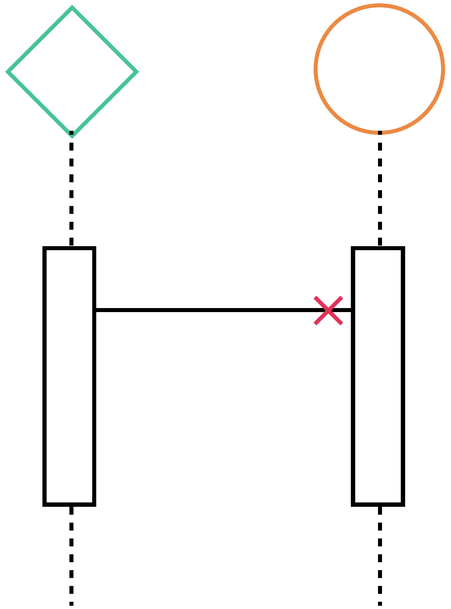
Тема занятия

1. Понятие хрупкости системы
2. Шаблоны повышения надёжности системы

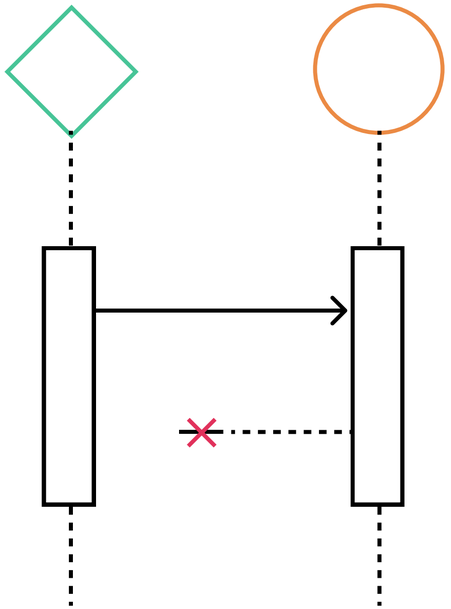
Шаблоны повышения надёжности системы

- Максимальное время ожидания — Timeout
- Повтор запроса — Retry
- Прерыватели цепи — Circuit Breaker
- Перегородки — Bulkhead
- Ограничитель количества запросов — Rate Limiter
- Изоляция
- Идемпотентность

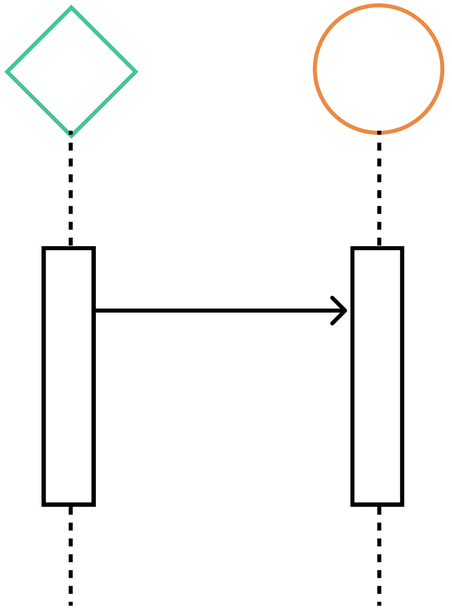
Timeouts



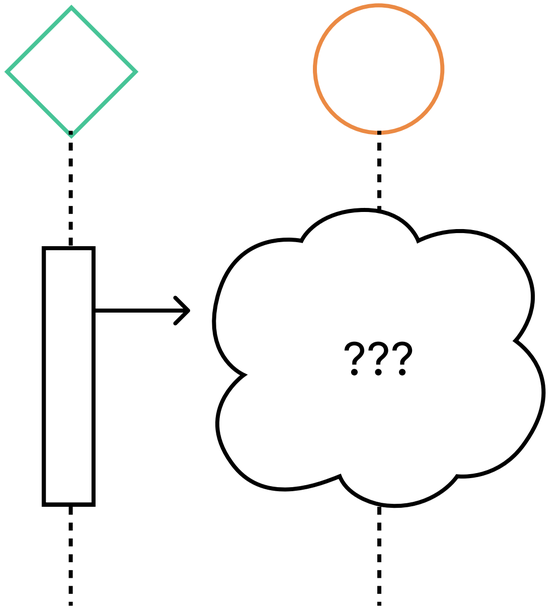
Запрос не дошёл
до получателя



Ответ не дошёл
от получателя

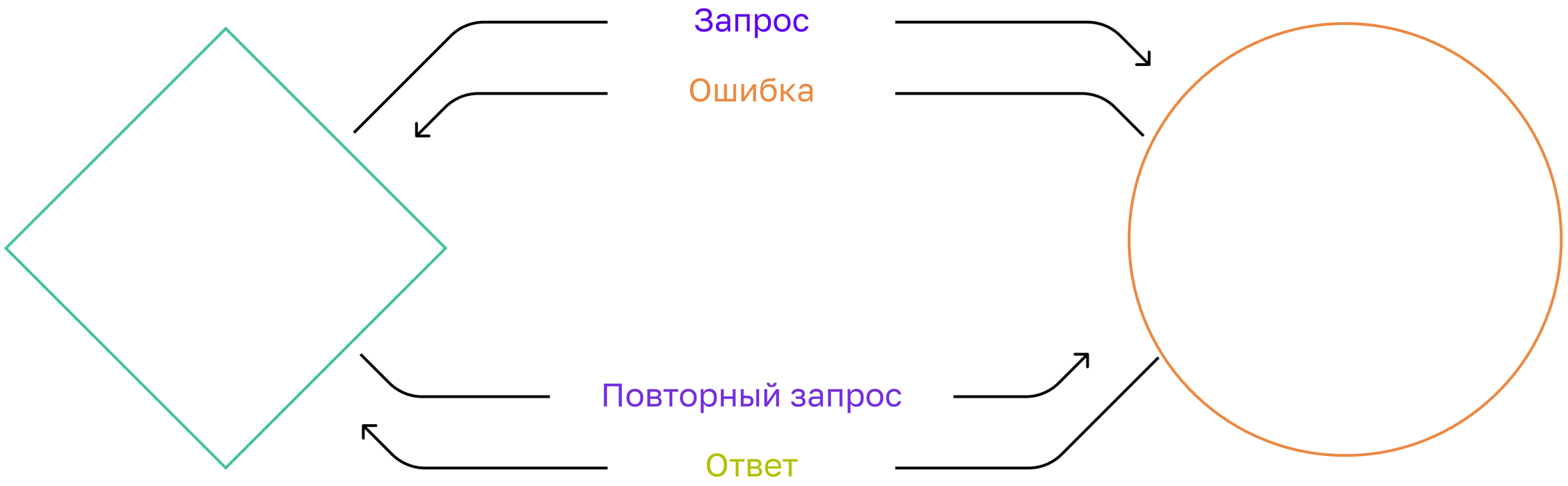


Получатель
не обработал запрос
или потерял его

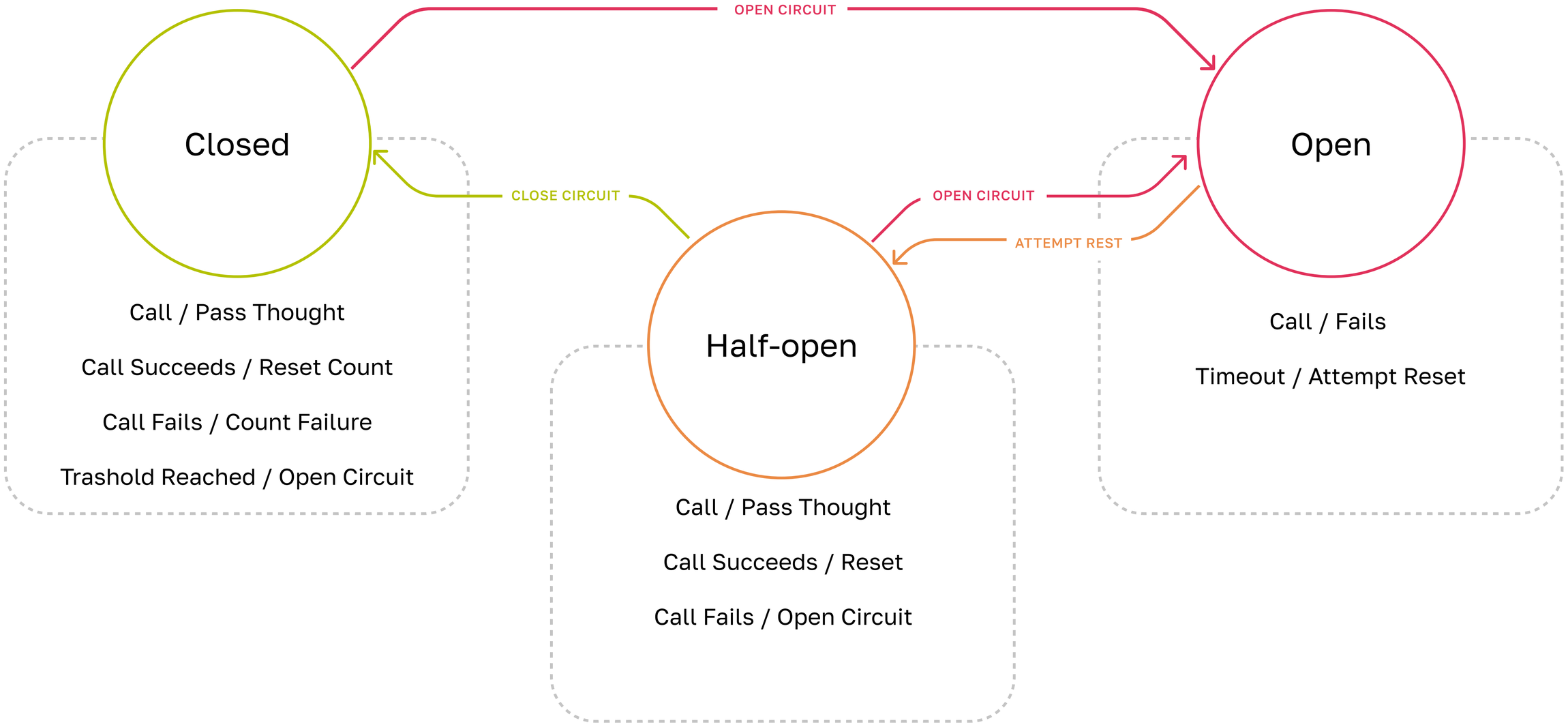


Неизвестно, был
ли ответ получен
и обработан

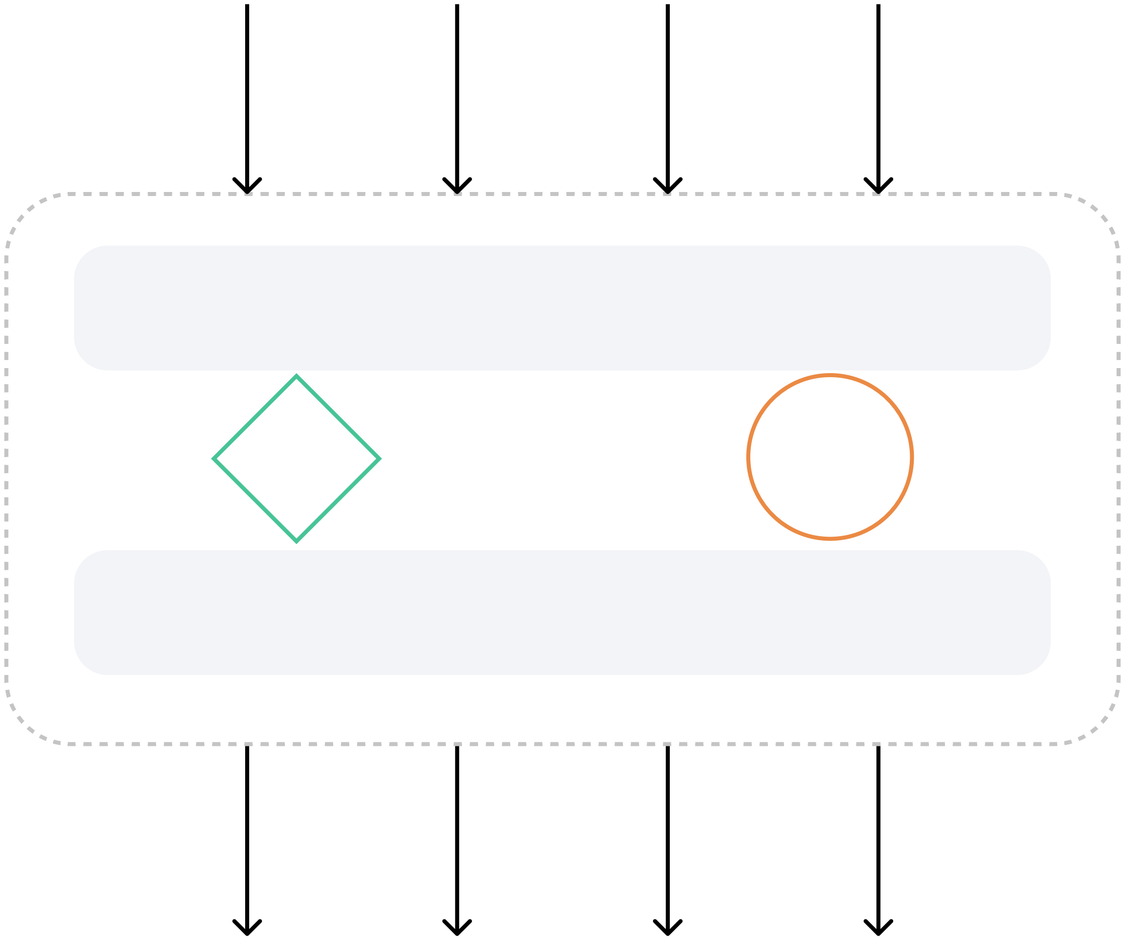
Повтор

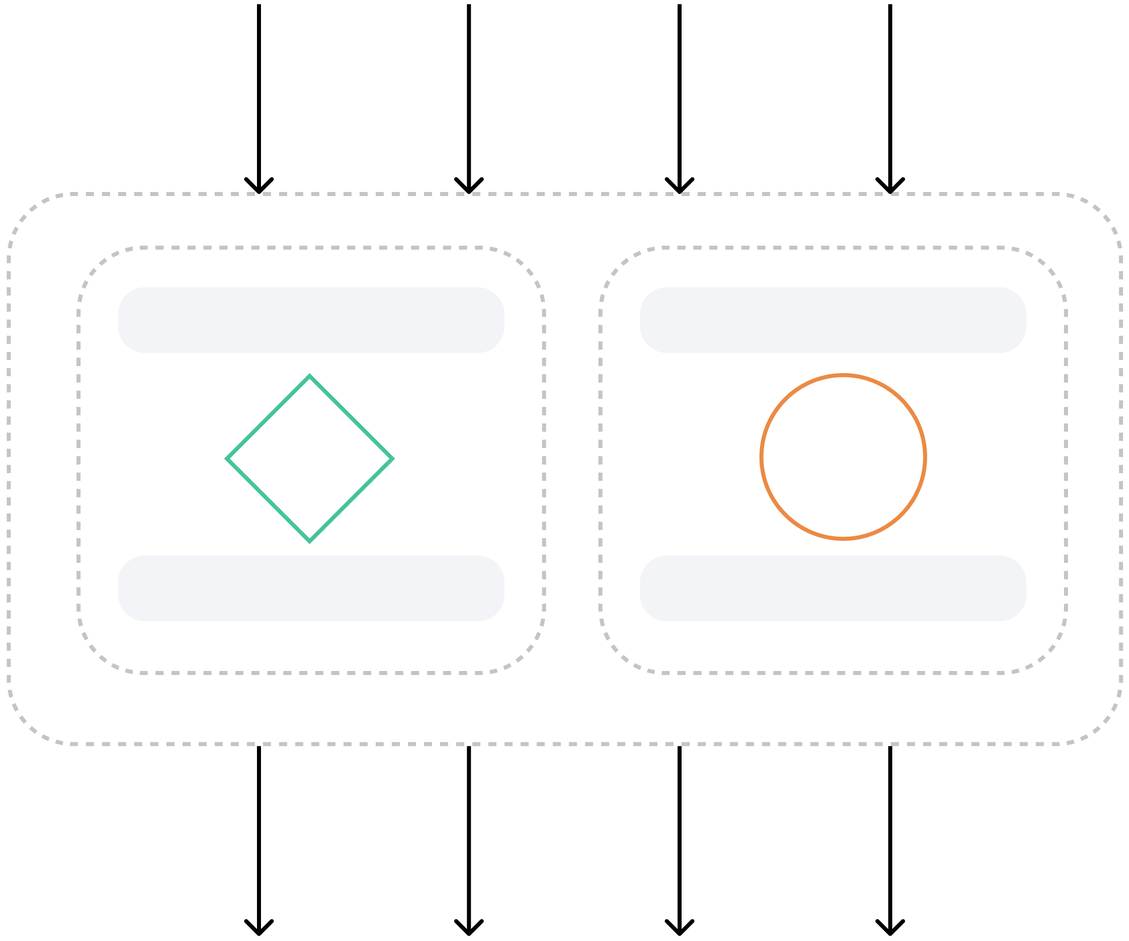


Прерыватель цепи

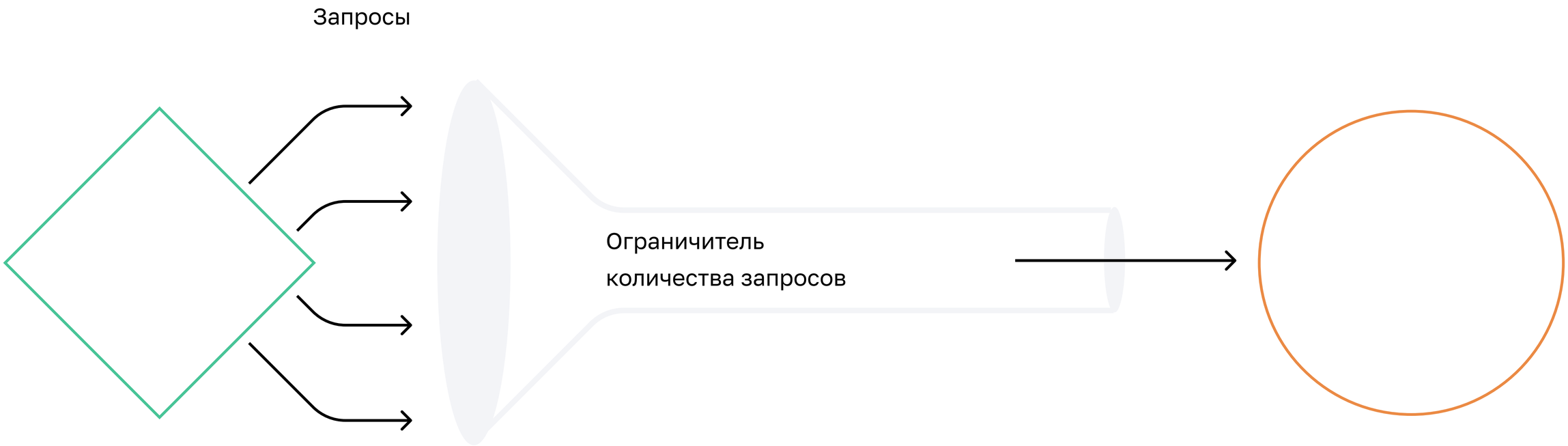


Перегородки





Ограничитель количества запросов



Итоги

- Применение различных подходов к обработке ошибок повышает устойчивость системы

Балансировка

Михаил Триполитов
Банк Открытие, архитектор решений

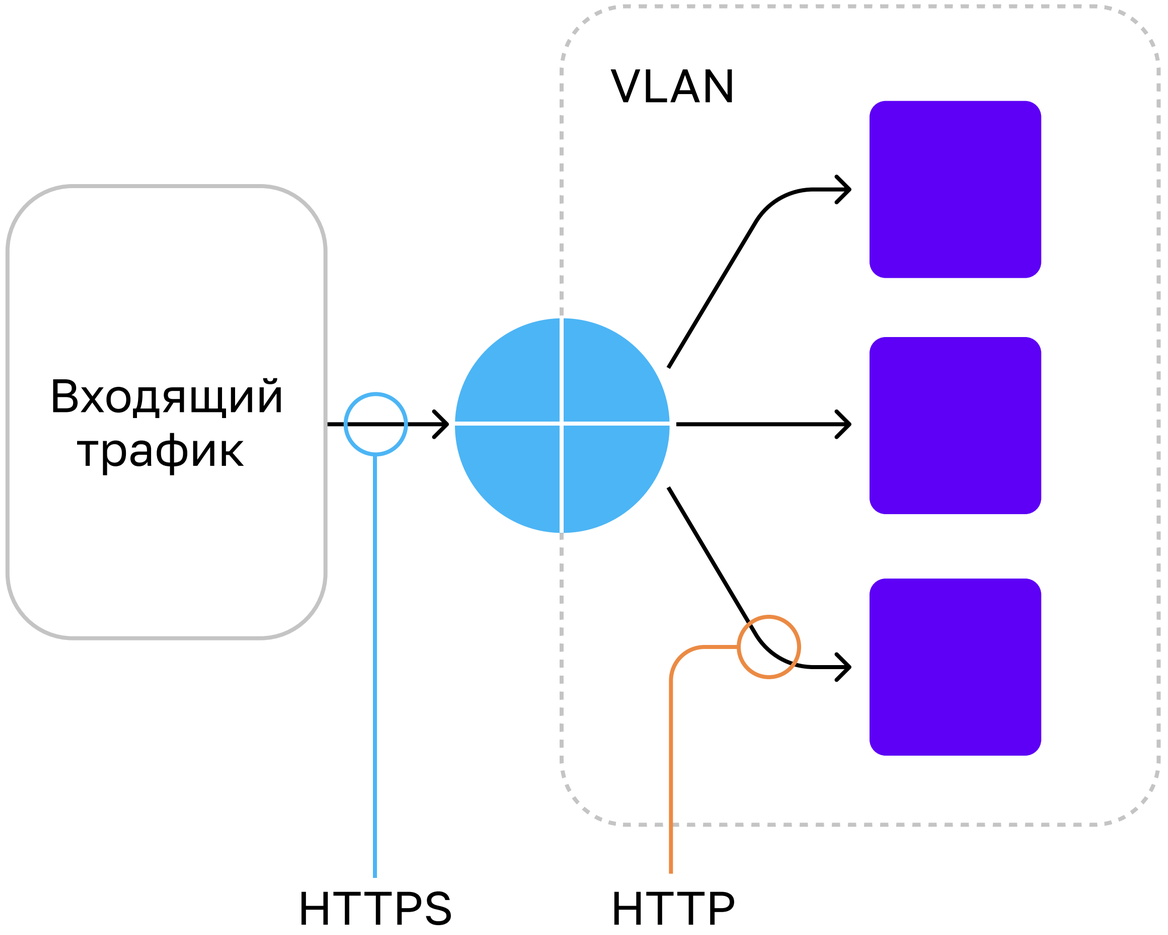
Тема занятия

1. Что такое балансировка нагрузки
2. Функциональность балансировщиков
3. Системы обработки задач

Балансировка

Дополнительные функции

- Терминация SSL
- Активация резервных серверов
- Ассиметричное распределение запросов
- Защита от DDOS-атак
- Gzip-сжатие HTTP
- Проверка работоспособности
- Кеширование HTTP
- Контентно-зависимое распределение запросов
- Аутентификация клиентов
- Фильтрация контента



Балансировщики

Балансировщики

- Nginx
- F5
- Kemp
- HAProxy
- Envoy

Облачные балансировщики

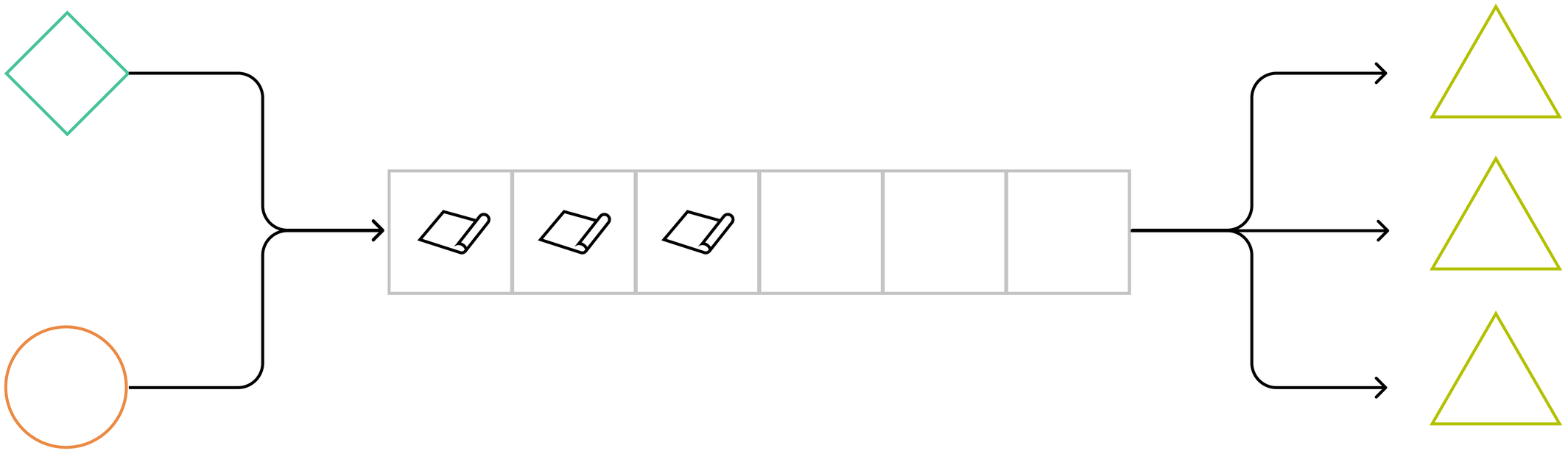
- AWS Elastic Load Balancer
- Google Cloud Load Balancing
- Azure Load Balancer
- Yandex Cloud Network Load Balancer

Системы обработки задач

Генераторы задач

Брокер сообщений

Конкурентные обработчики задач



Итоги

- Балансировка нагрузки позволяет избежать единой точки отказа и распределить нагрузку по нескольким серверам
- Нагрузку можно балансировать как с помощью балансировщика запросов, так и при помощи системы распределения задач

Кеширование

Михаил Триполитов
Банк Открытие, архитектор решений

Тема занятия

1. Кеширование
2. Способы кеширования

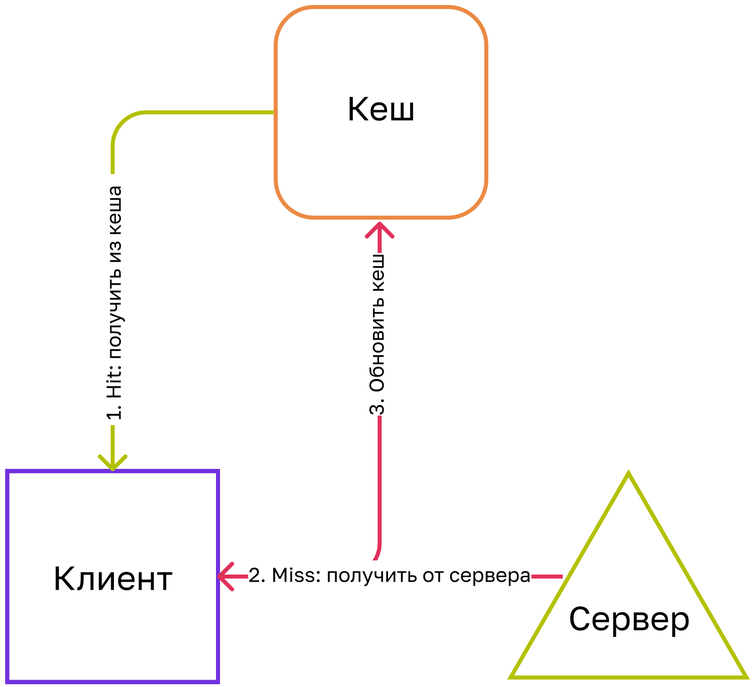
Кеширование

! **Кеширование** — наиболее распространённый способ повышения производительности

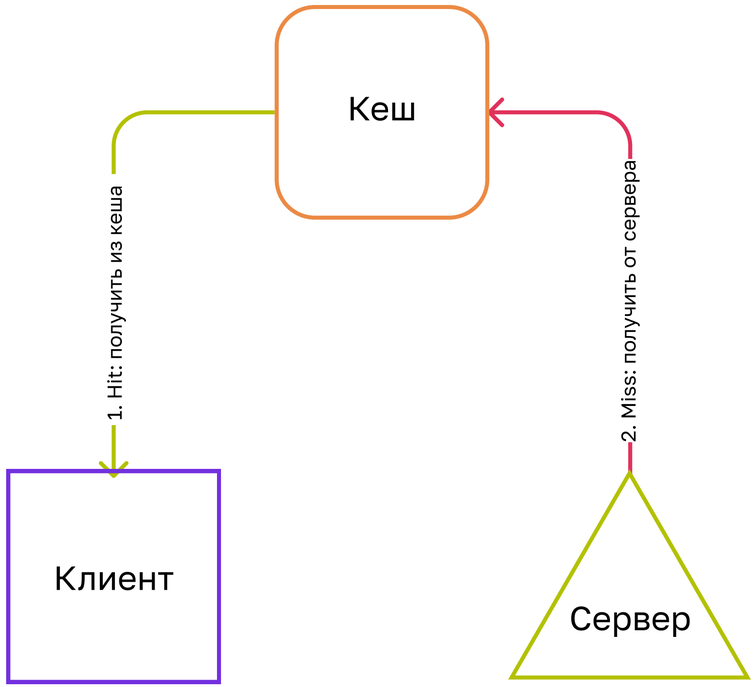
Кеширование

Кеширование — наиболее распространённый способ повышения производительности

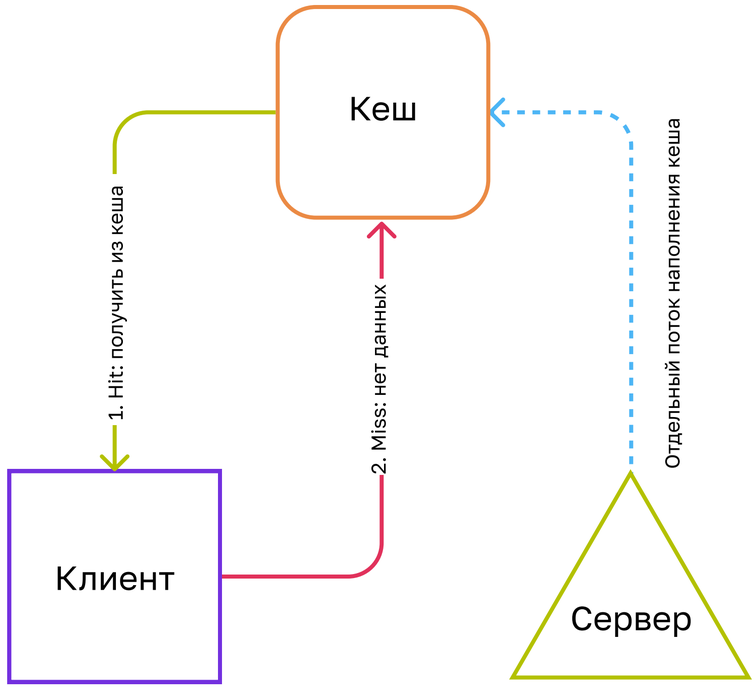
Клиентский кеш



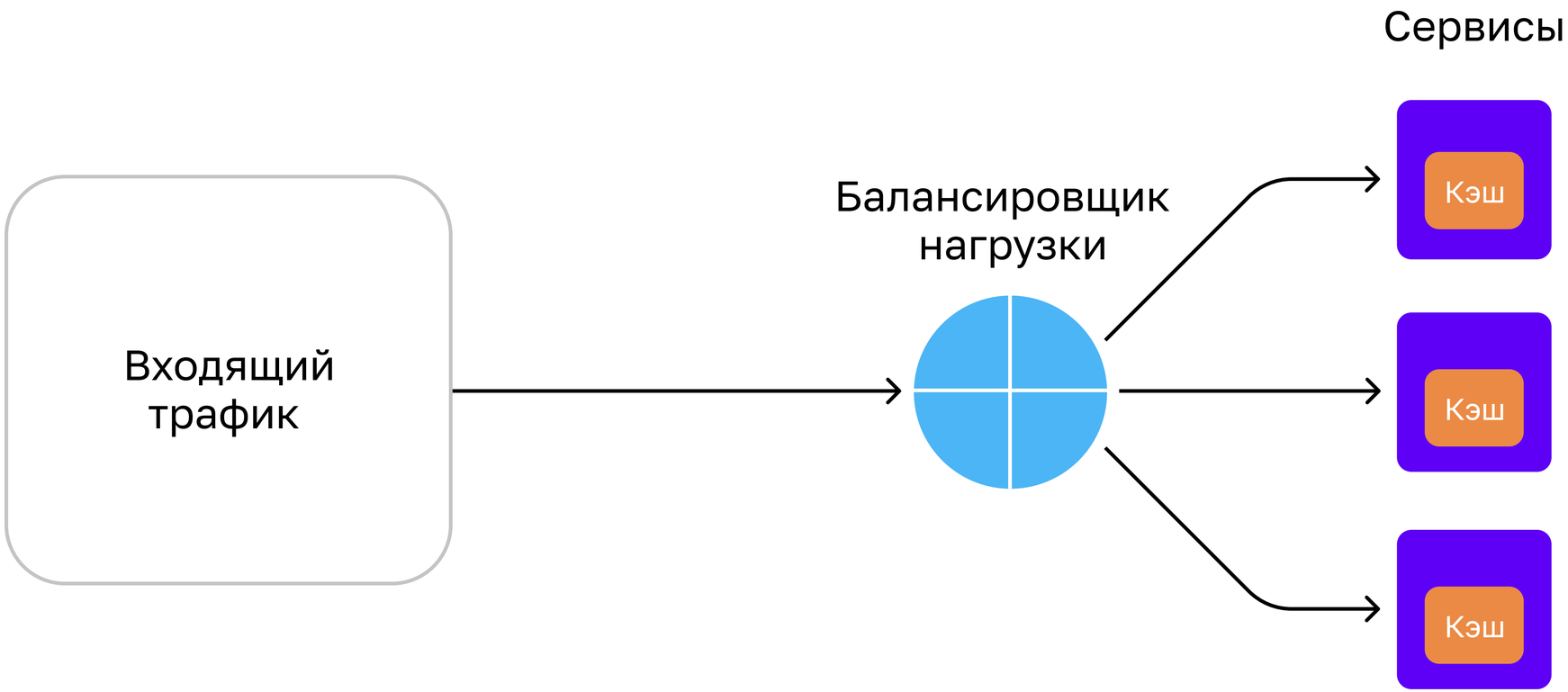
Сквозной кеш



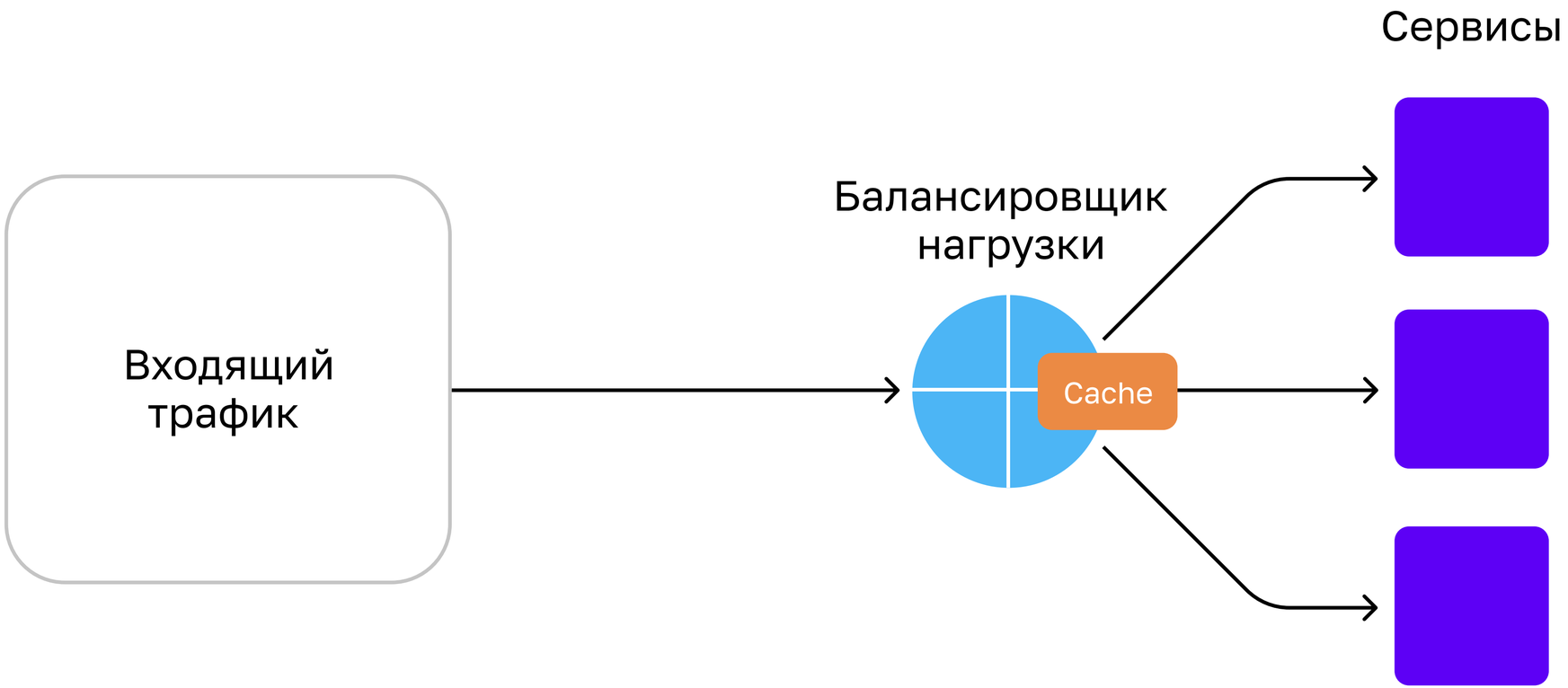
Прогретый кеш



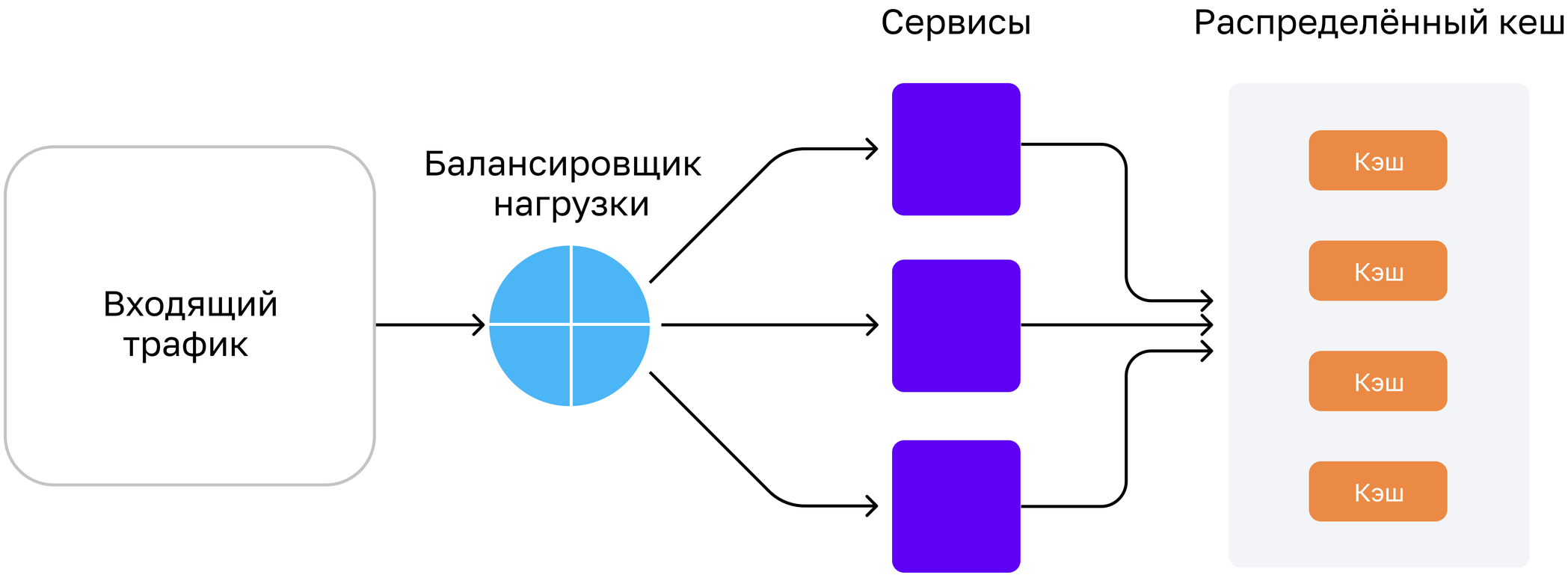
Кеш встроен в сервис



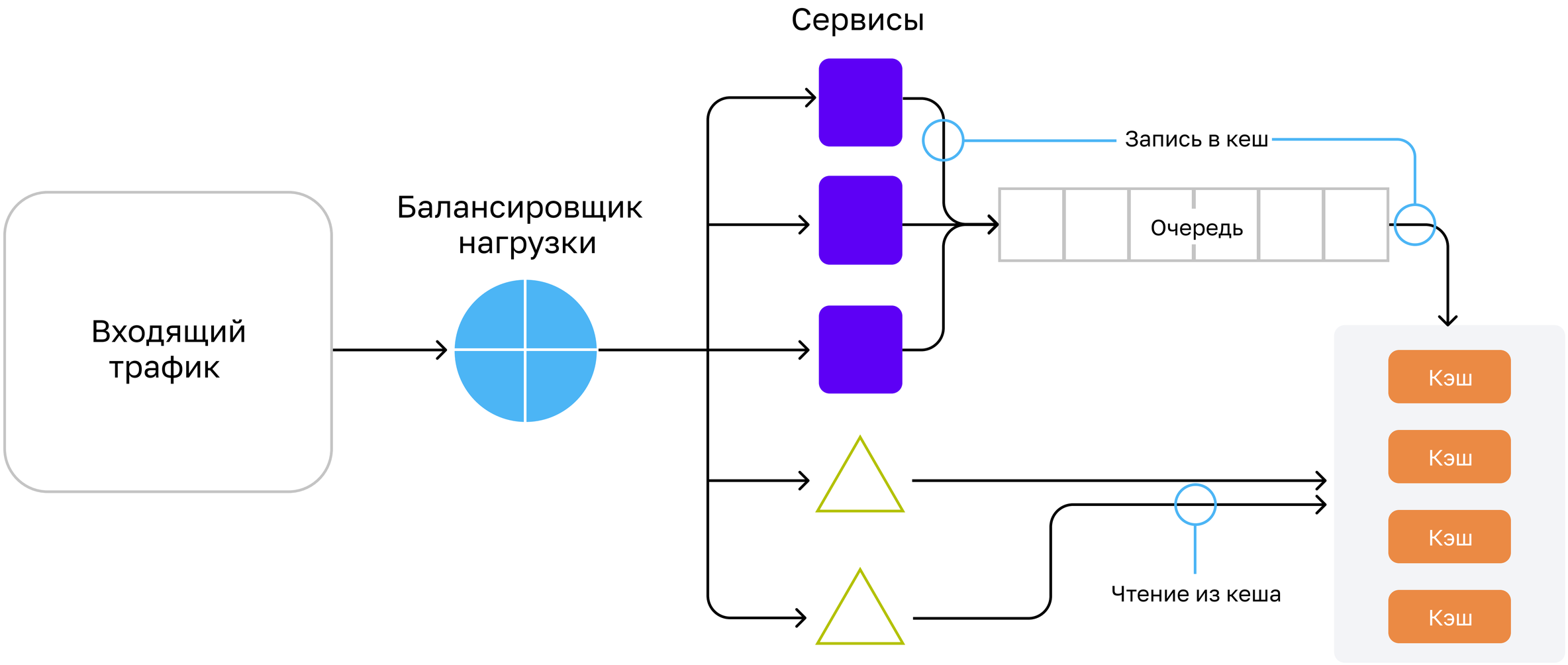
Кеширующий балансировщик



Распределённый кеш



Прогретый кеш



In-Memory кеш

- Hazelcast
- Redis
- Memchached
- Tarantool

Итоги

- Кеширование — наиболее распространённый способ повышения производительности приложения

Автомасштабирование

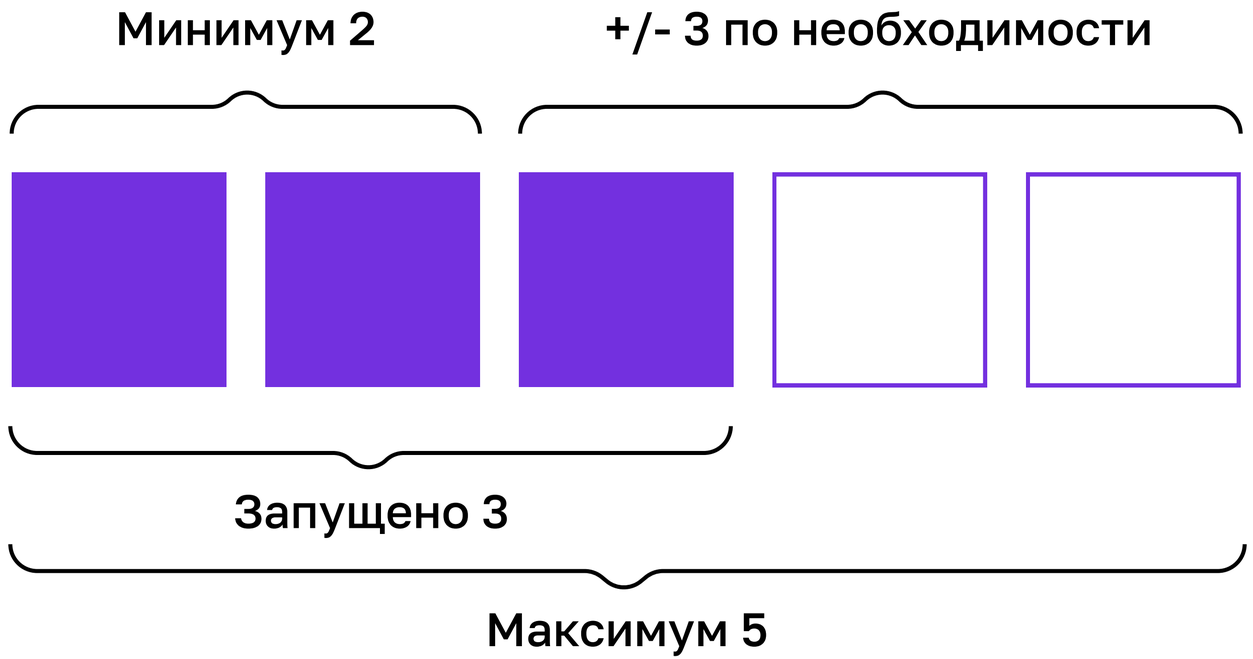
Михаил Триполитов
Банк Открытие, архитектор решений

Тема занятия

1. Кеширование
2. Способы кеширования
3. Подходы к автомасштабированию

Автоматическое масштабирование

Процесс динамического выделения ресурсов для удовлетворения требований производительности: **при увеличении нагрузки** выделяются дополнительные ресурсы для обеспечения соглашения об уровне обслуживания (SLA)



Необходимые компоненты

- Мониторинг приложений и инфраструктуры
- Логика принятия решений
- Управление ресурсами
- Тестирование процедуры автомасштабирования

Подходы

- Динамическое масштабирование на основе метрик приложения или инфраструктуры:
  - среднее значение CPU
  - общее количество запросов
  - длительность обработки задачи
  - количество запросов, завершившихся ошибкой
- Масштабирование по расписанию
- Предсказательное масштабирование

Автомасштабирование

- Kubernetes Horizontal Pod Autoscaler
- Amazon EC2 Auto Scaling
- Azure Monitor
- Google Compute Engine Autoscaling
- Yandex Cloud Cluster-Autoscaler

Итоги

- Автоматическое масштабирование позволяет снизить расходы на эксплуатацию системы
- Автоматическое масштабирование проще всего применять в областных хостингах

Service Discovery & Service Mesh

Михаил Триполитов
Банк Открытие, архитектор решений

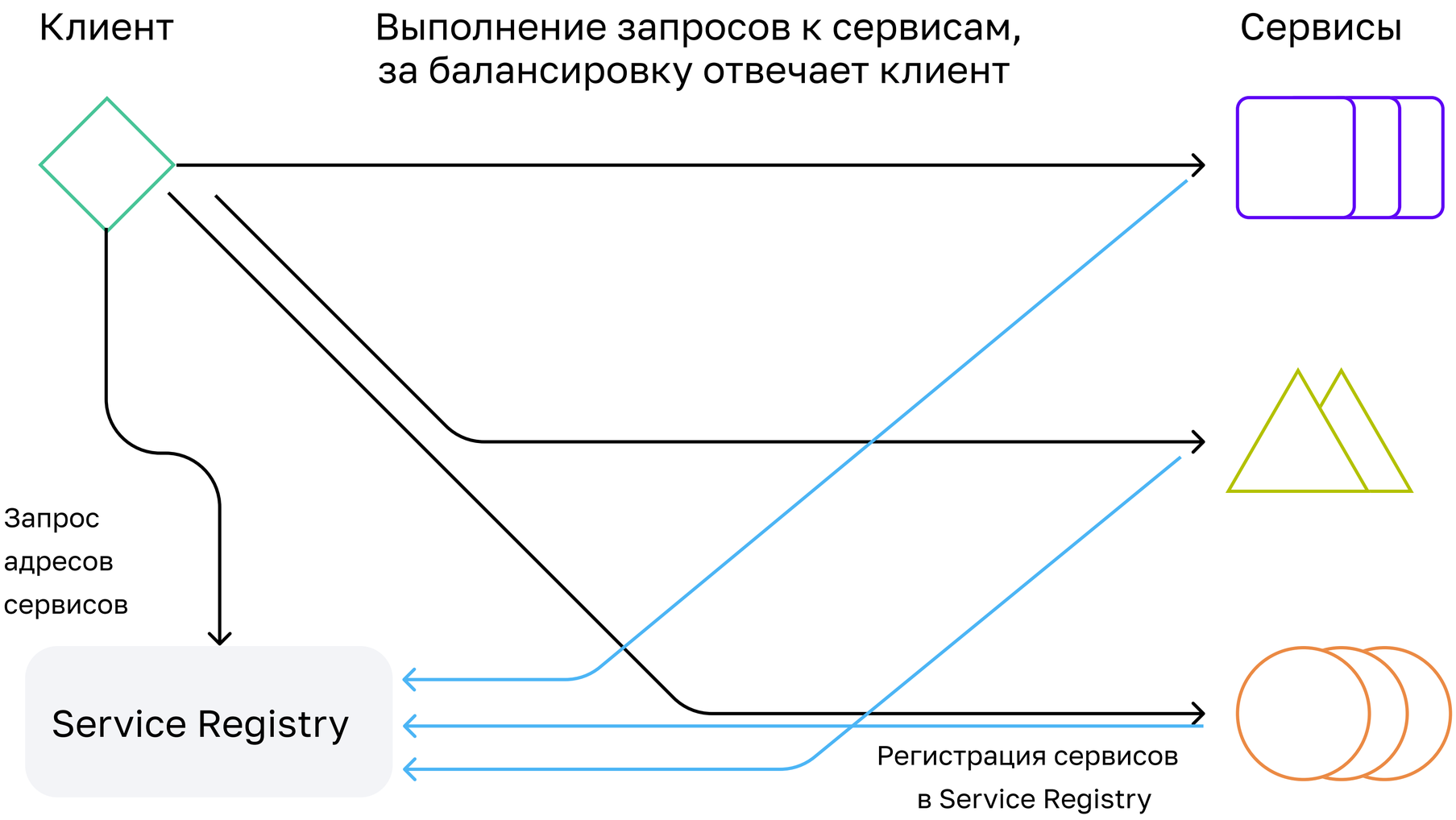
Тема занятия

1. Service Discovery
2. Возможные реализации
3. Задачи Service Mesh

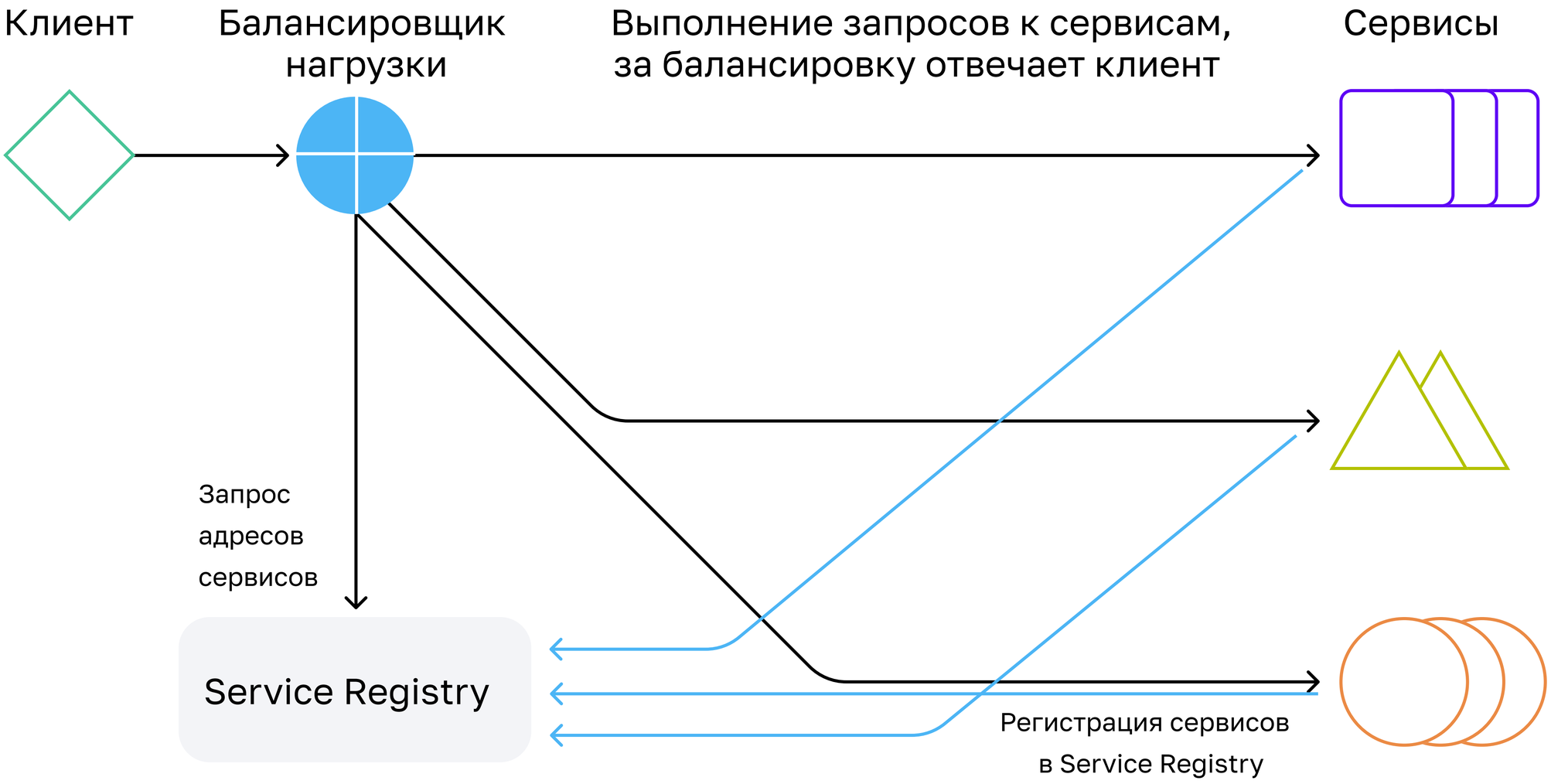
Service Discovery

- **Настройки**
  У клиентов в конфигурации IP-адреса сервисов, с которыми ему нужно взаимодействовать
- **DNS**
  У каждого сервиса есть DNS-имя, за которым несколько IP-адресов его запущенных экземпляров
- **Service Registry**
  Специальное программное обеспечение позволяет сервисам при старте зарегистрировать адреса запускаемого экземпляра, а клиентам получить эти адреса по имени

Service Registry



Service Registry



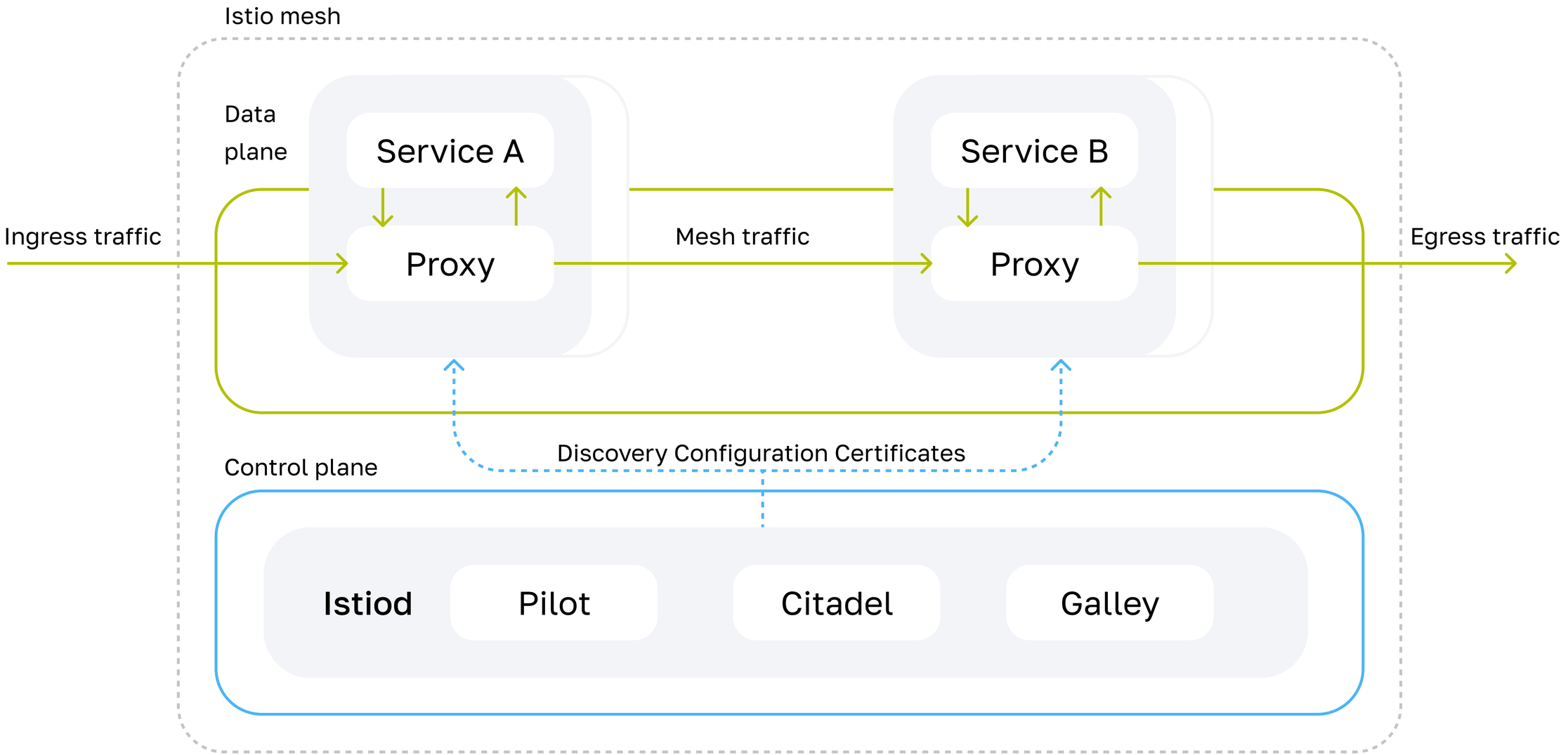
Service Registry

- Zookeeper
- Consul
- Eureka
- Самодельный

Задачи Service Mesh

- **Гибкость настройки** взаимодействия сервисов на уровне инфраструктуры, управление трафиком, балансировка
- **Унифицированный мониторинг** большой системы
- **Повышение безопасности:** отсутствие риска перехвата трафика, полный контроль над сетью, исключение угроз при работе в нескольких дата-центрах или публичных облаках
- **Упрощение реализации сложных шаблонов развёртывания приложений:** Blue/Green, Canary, A/B testing
- **Организация взаимодействия сервисов в мультикластерной среде**

Принцип работы Service Mesh на примере Istio



Service Mesh

- Istio
- Linkerd
- Consul Connect
- Kuma
- Maesh
- OpenShift Service Mesh
- AWS App Mesh

Итоги

- При разделении системы на большое количество сервисов необходим механизм Service Discovery
- Service Mesh позволяет решить проблемы Service Discovery в контейнерных средах