Надежность в распределенных системах

Олег Анастасьев
@m0nstermind
oa@ok.ru

В Одноклассниках

1. Абсолютно надежная сеть
2. Мизерная сетевая задержка
3. Практически безлимитная пропускная способность
4. Полностью однородна
5. Изменения топологии сети незаметны
6. Полностью защищена
7. Управляется одним администратором
8. Транспортировка данных почти ничего не стоит

Заблуждения разработчиков распределенных систем

1. Абсолютно надежная сеть
2. Мизерная сетевая задержка
3. Практически безлимитная пропускная способность
4. Полностью однородна
5. Изменение топологии сети незаметны
6. Полностью защищена
7. Управляется одним администратором
8. Транспортировка данных почти ничего не стоит

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

[Peter Deutsch, 1994; James Gosling 1997]

В Одноклассниках

ЦОД

типов
сервисов

серверов

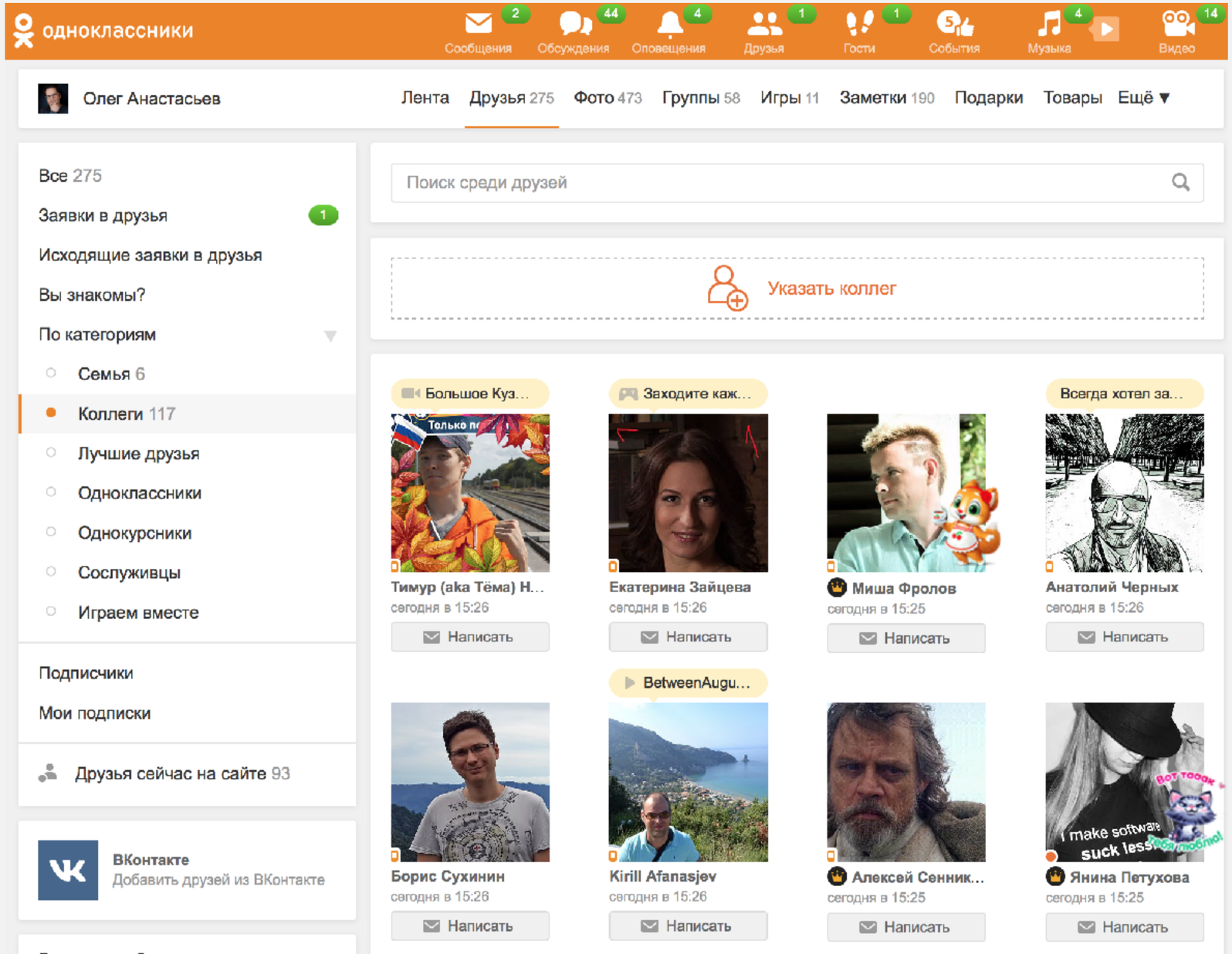
инженеры

сетевики

админы

разработчики

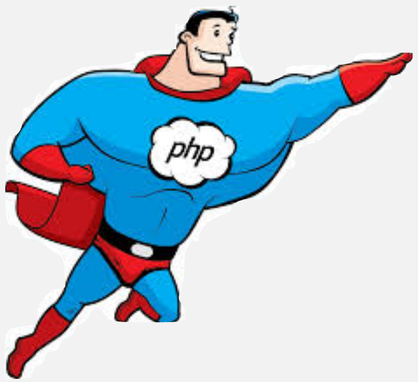
Страница друзей



1. Получить список друзей
2. Применить фильтр
3. Исключить ЧС
4. Получить профили
5. Отсортировать
6. Получить наклейки
7. Посчитать счетчики

Простой способ

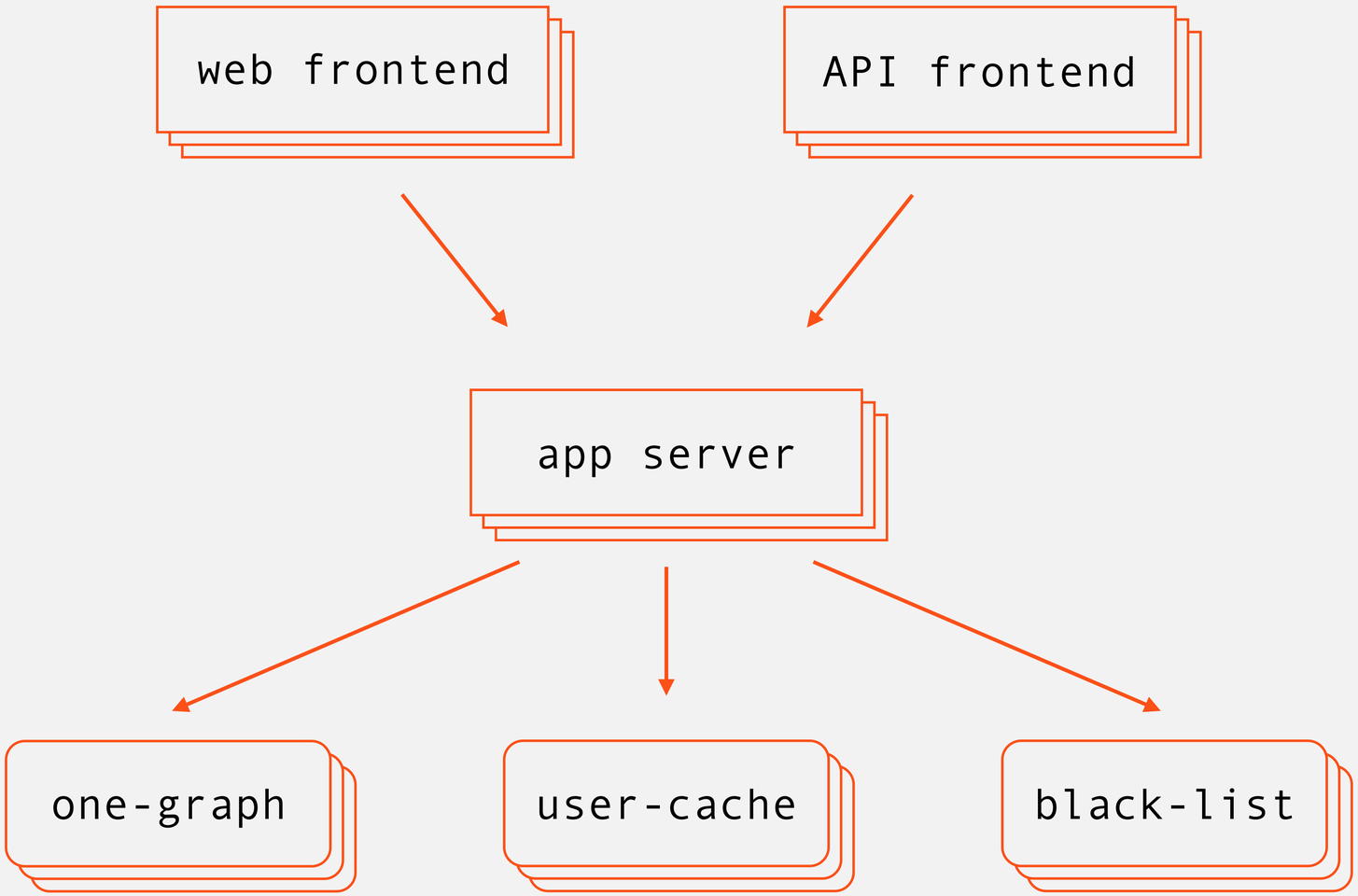
```
SELECT * FROM friendlist f, users u
    JOIN ON f.vertexId=u.userId
    WHERE u.userId=? AND f.kind=?
AND NOT EXISTS( SELECT * FROM blacklist …)
    …
```



Простой способ не работает

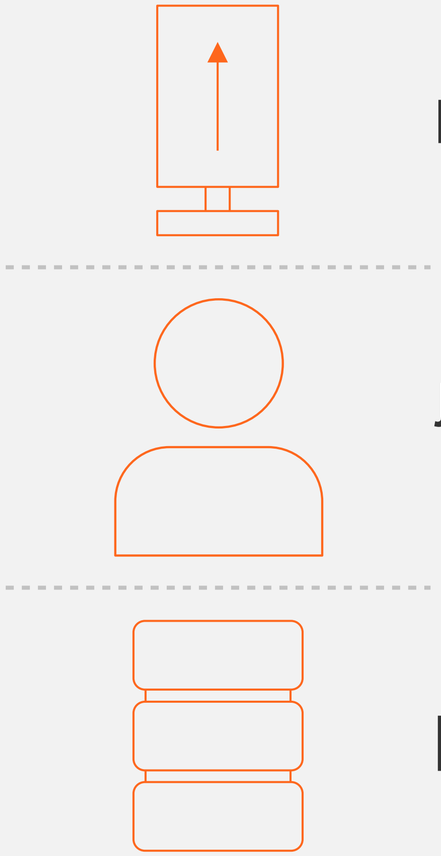
- Дружбы
  - 12 млрд. связей, 300GB
  - 500 000 запросов в сек.
- Профили пользователей
  - > 350 млн. штук,
  - 3.5 млн. запросов в сек.
  - 50 Gbit

Работает так



(микро) сервисы

Анатомия (микро) сервиса



Ремотный интерфейс

Логика , Кэши

[Локальное хранилище]

У нас есть OpenSource для этого:

https://github.com/odnoklassniki/one-nio

Почему так тут: *Класс!ная кассандра*

https://www.youtube.com/watch?v=k2efjgRxMp8

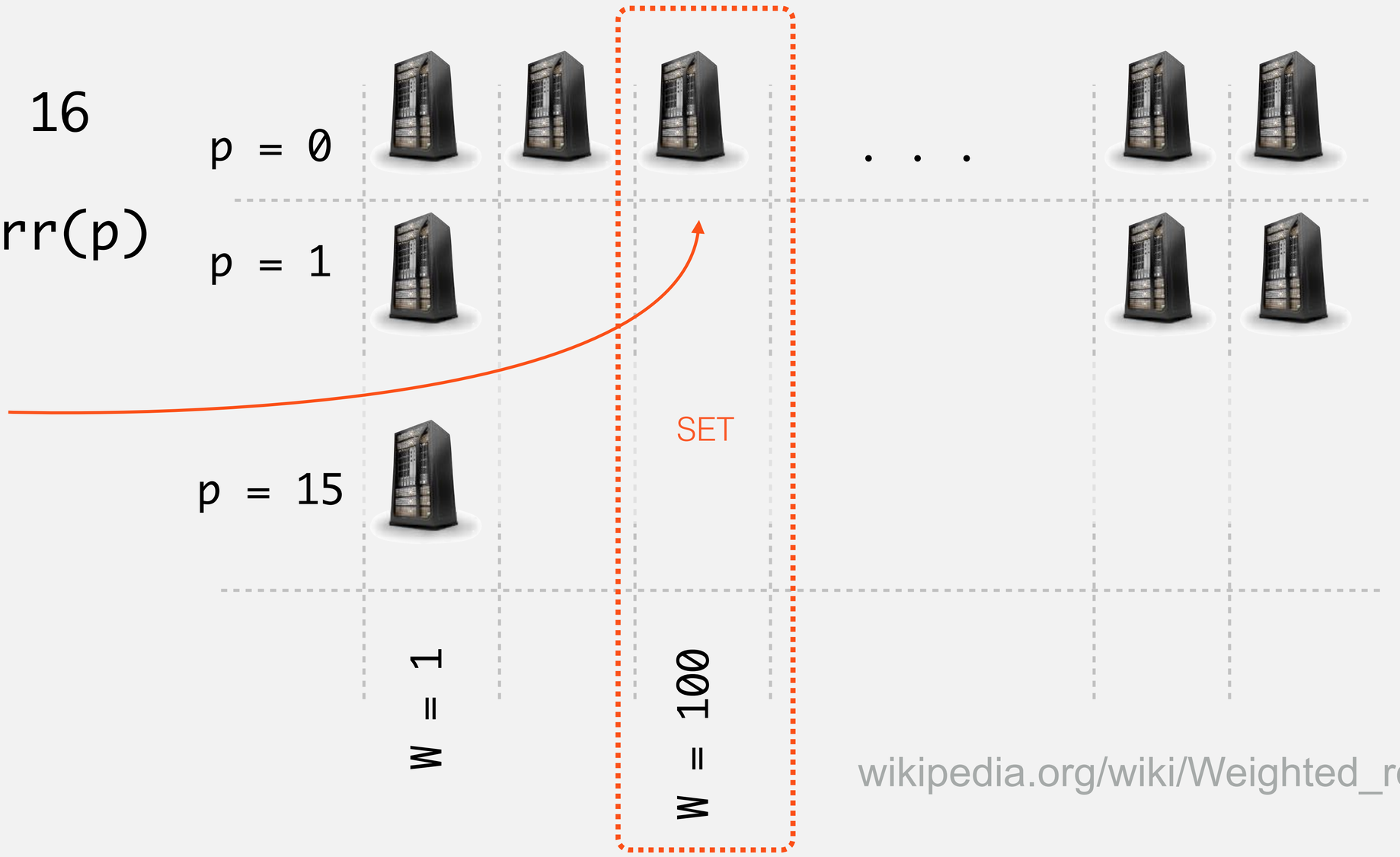
Взвешенный квадрат

```
p = id % 16

node = wrr(p)
```

1 ms

10 сек



wikipedia.org/wiki/Weighted_round_robin

split & merge

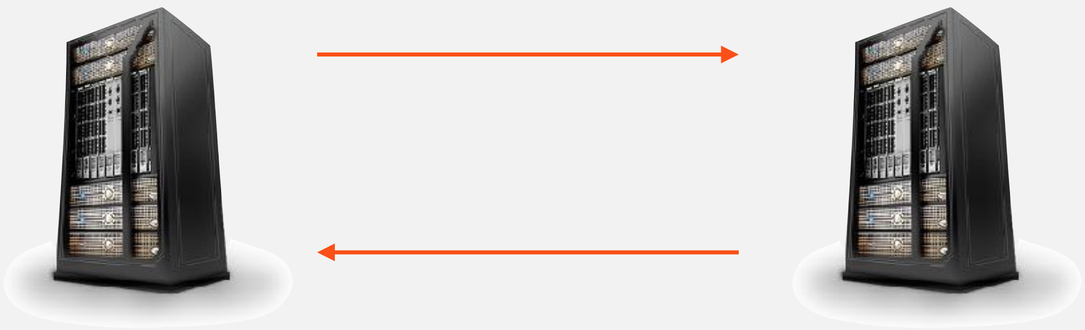
```
split ( ids by p )
      -> ids0, ids1
```

ids1

ids0

```
users = merge (users0, users1)
```

Что может пойти не так ?



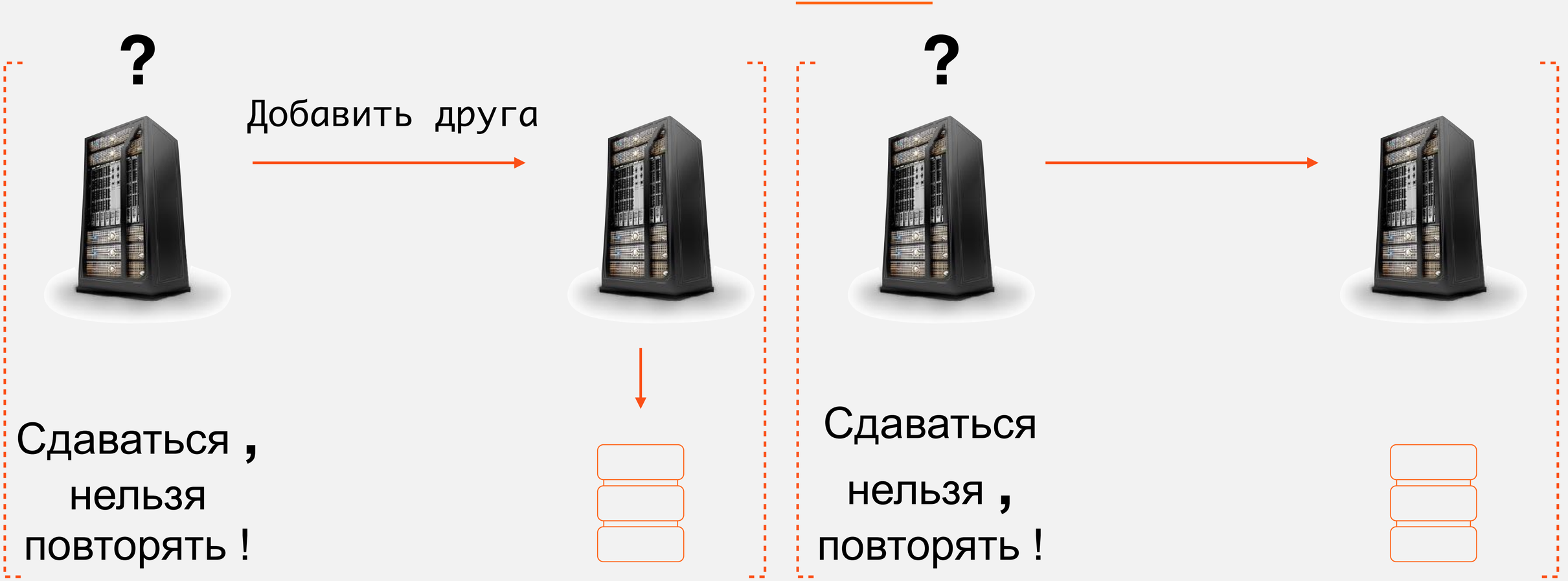
1. Пропажа клиента
2. Пропажа сервера
3. Потеря исходящего сообщения
4. Потеря входящего сообщения
5. Таймаут сервера
6. Неправильный ответ
7. Произвольный отказ

Отказы

Что делать с отказами ?

- Игнорировать
- Отказ можно ~~предотвратить~~ скрыть
- Отказ произойдет **обязательно**.
- Ключ к скрытию отказов — избыточность:
  - Информации (коды защиты от ошибок)
  - Железа (резервирование, реплики, дублирующие схемы)
  - Времени (транзакции, retries)

Что сервер сделал ?



Был ли друг добавлен ?

- Со стороны клиента — неизвестно
- Что клиент может сделать ?
  - Не давать никаких гарантий
  - Никогда не повторять запрос. Максимум 1 раз. At Most Once.
  - Всегда повторять запрос. По меньшей мере 1 раз. At Least Once.
  - ~~Ровно 1 раз. Exactly Once~~

Добавляем друга. Часть 1

1. Транзакция в ACID хранилище

- есть мастер, успех однозначен (или проходит, или нет)
- возможен атомарный откат

https://ru.wikipedia.org/wiki/ACID

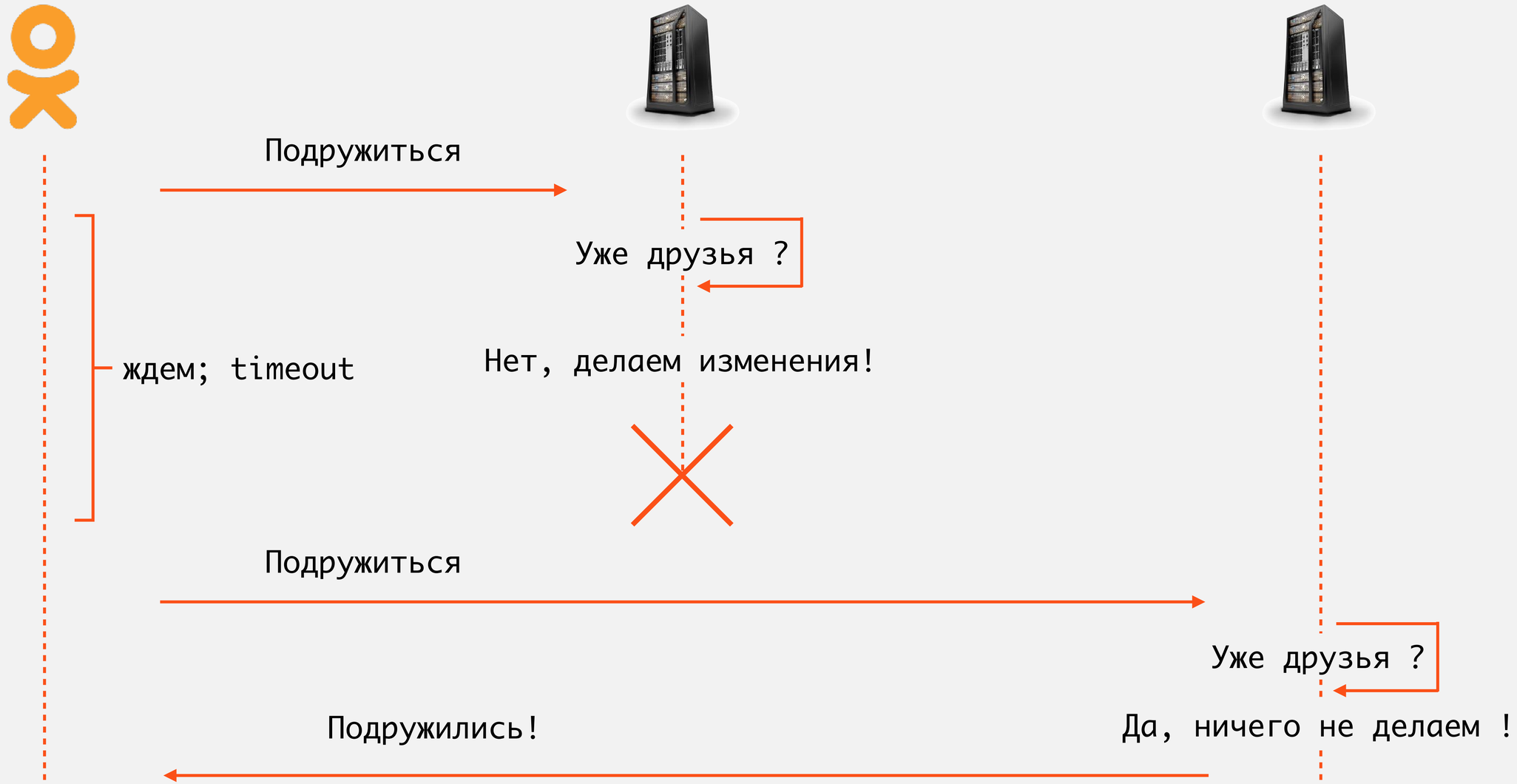
Идемпотентность

- Операция применима повторно с тем же результатом
  - Чтение, Set.add(), Math.max(x,y)
  - CRDT https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type 
  - Упорядоченное атомарное изменение с контролем дубликата

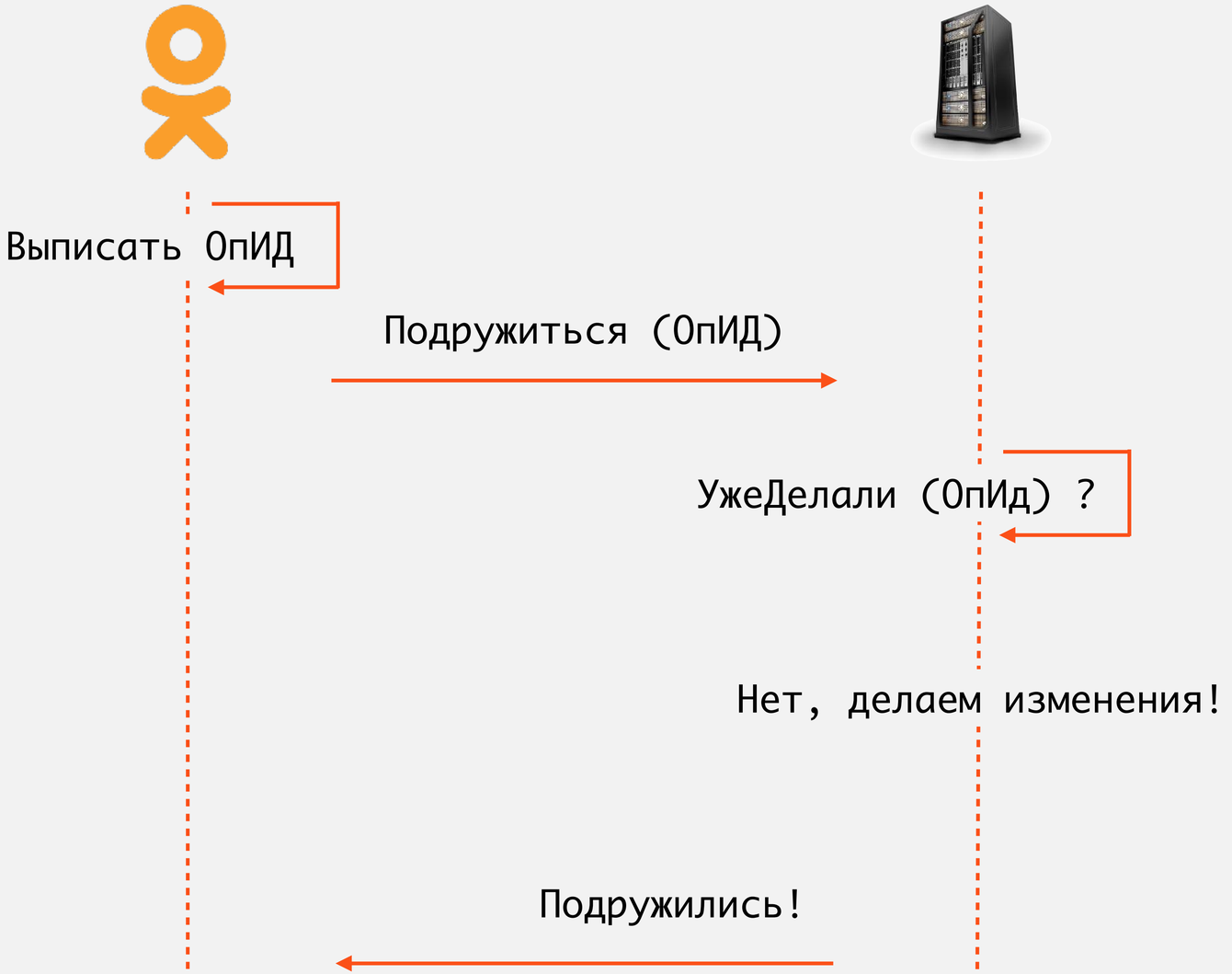
Только для Идемпотентных операций
можно применять стратегию
“всегда повторять попытку”

https://ru.wikipedia.org/wiki/Идемпотентность

Идемпотентность в ACID хранилище



Идемпотентность через секвенсинг



Примеры ОпИД:

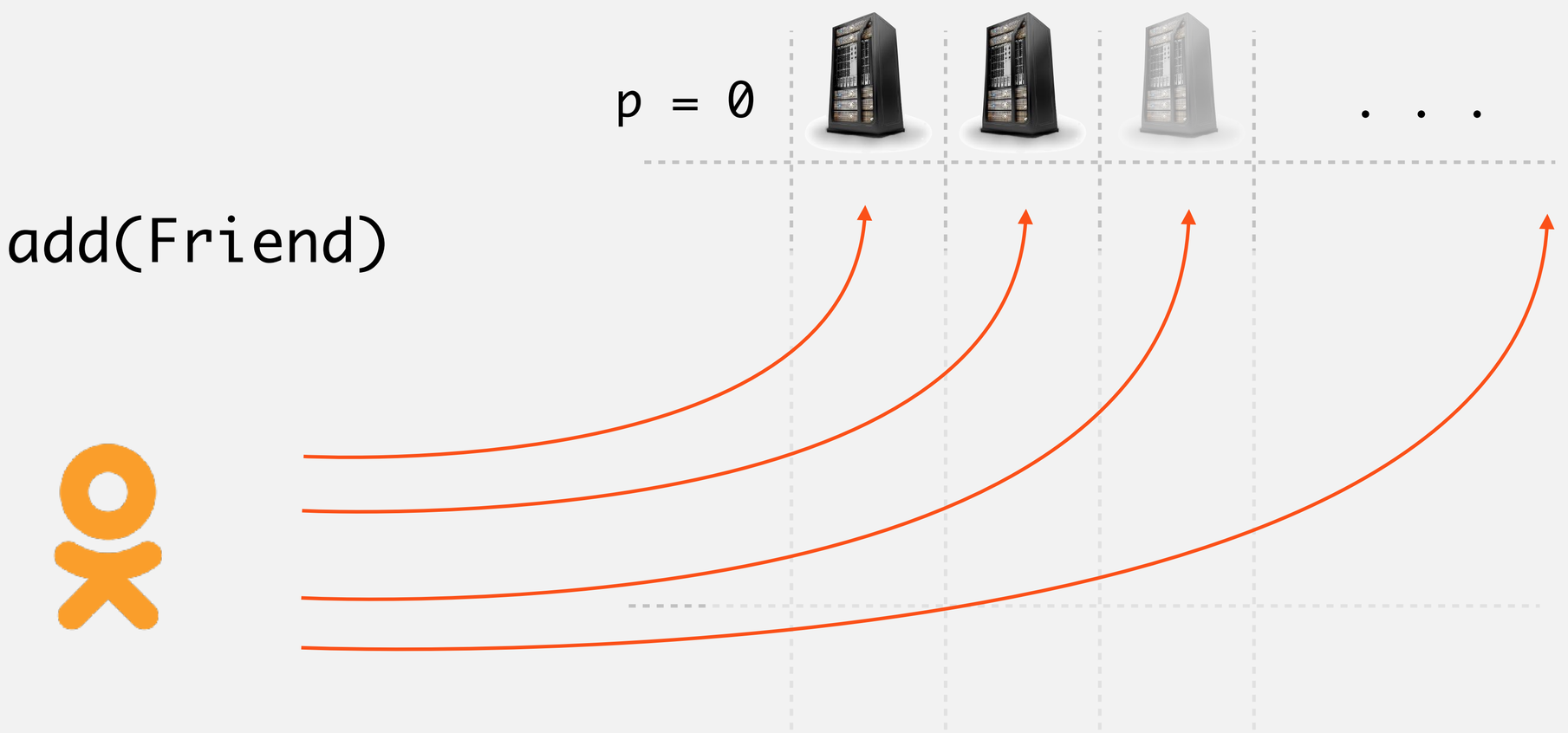
- `OpId+=1`
- `OpId=currentTimeMillis()`
- `OpId=UUID`

https://en.wikipedia.org/wiki/Universally_unique_identifier

Добавляем друга. Часть 2

1. Транзакция в ACID хранилище
   - есть мастер, успех однозначен (или проходит, или нет)
   - возможен атомарный откат
2. Обновление информации в кэшах
   - много реплик, мастера нет
   - атомарного отката нет: возможны частичные отказы

Нотификация реплик кэшей



Повторять бессмысленно

Но без повтора данные реплики рассинхронизируются

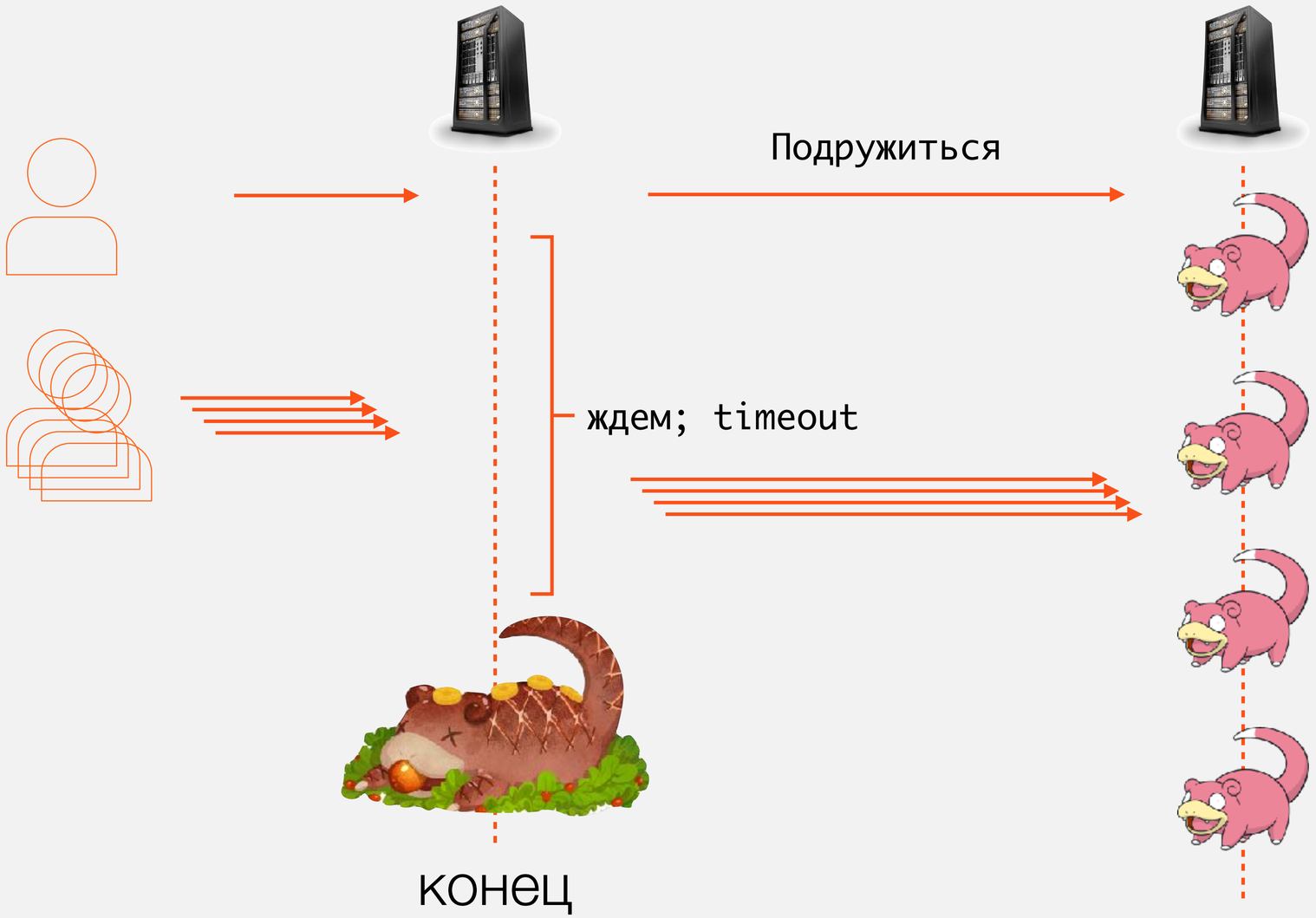
Синхронизируем кэш через БД

- Процесс синхронизации данных
  - Непрерывно читает изменения из транзакционного хранилища

    ```
    SELECT * FROM users WHERE modified > ?
    ```

  - Применяет их в память кэша
- Загружает изменения при старте ноды
- Повтор — не нужен

Смерть через таймаут



Вывод из ротации

1. Клиенты перестают обращаться к серверу

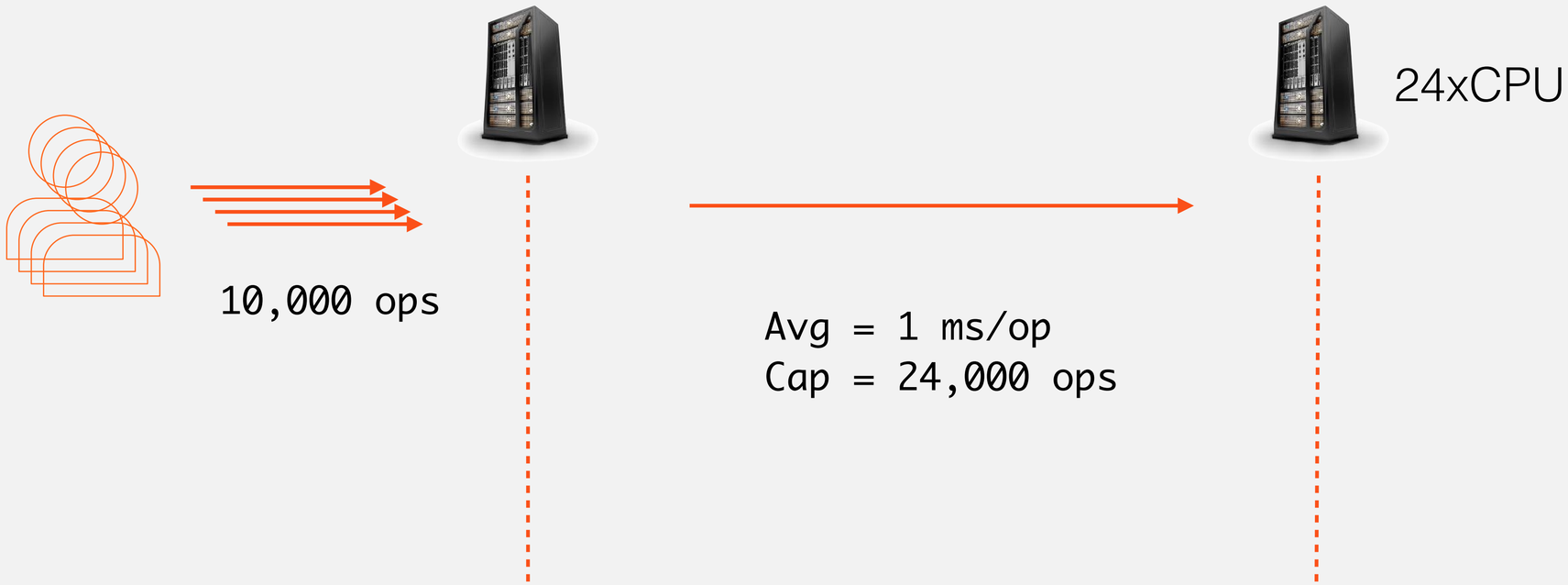
   После X последовательных отказов за последнюю секунду

2. Клиенты мониторят доступность сервера

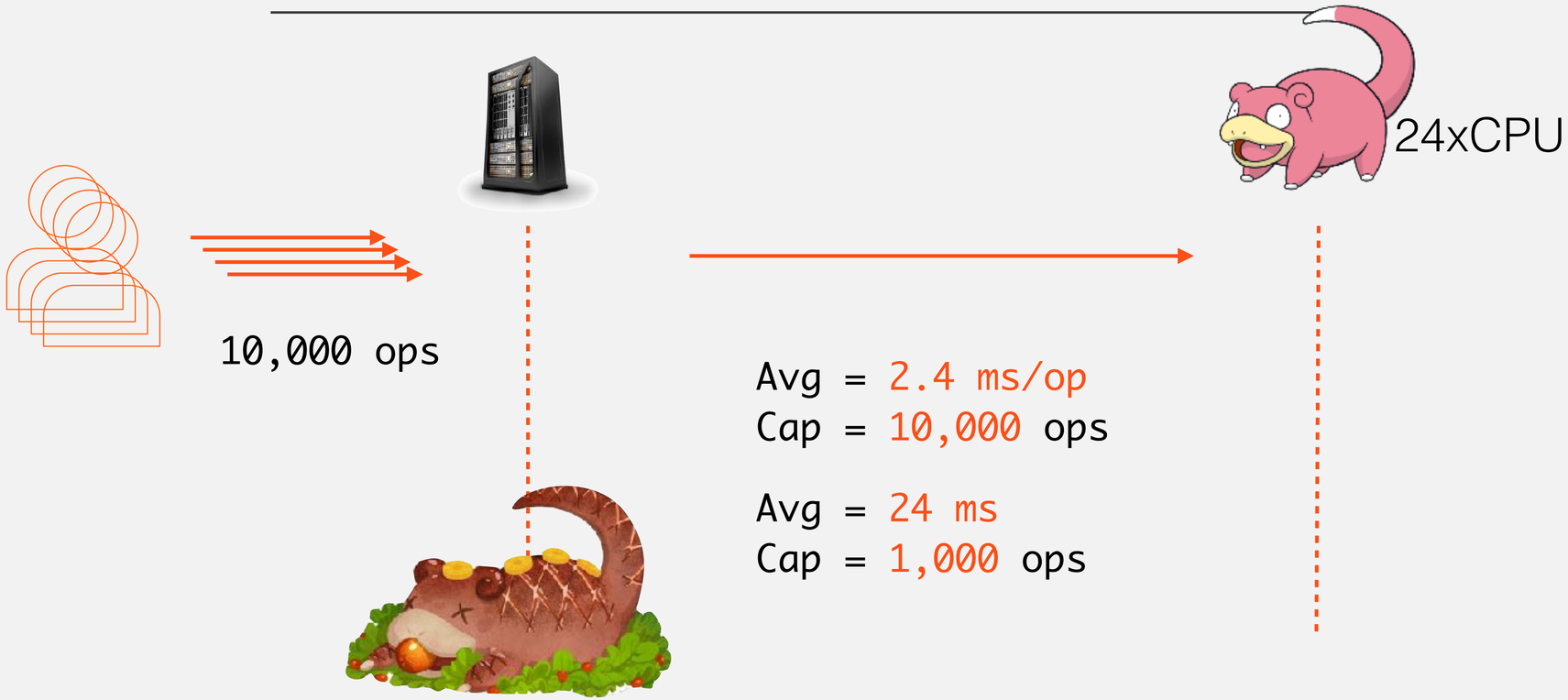
   В фоне, раз в минуту

3. И возвращают его в ротацию

Смерть через торможение



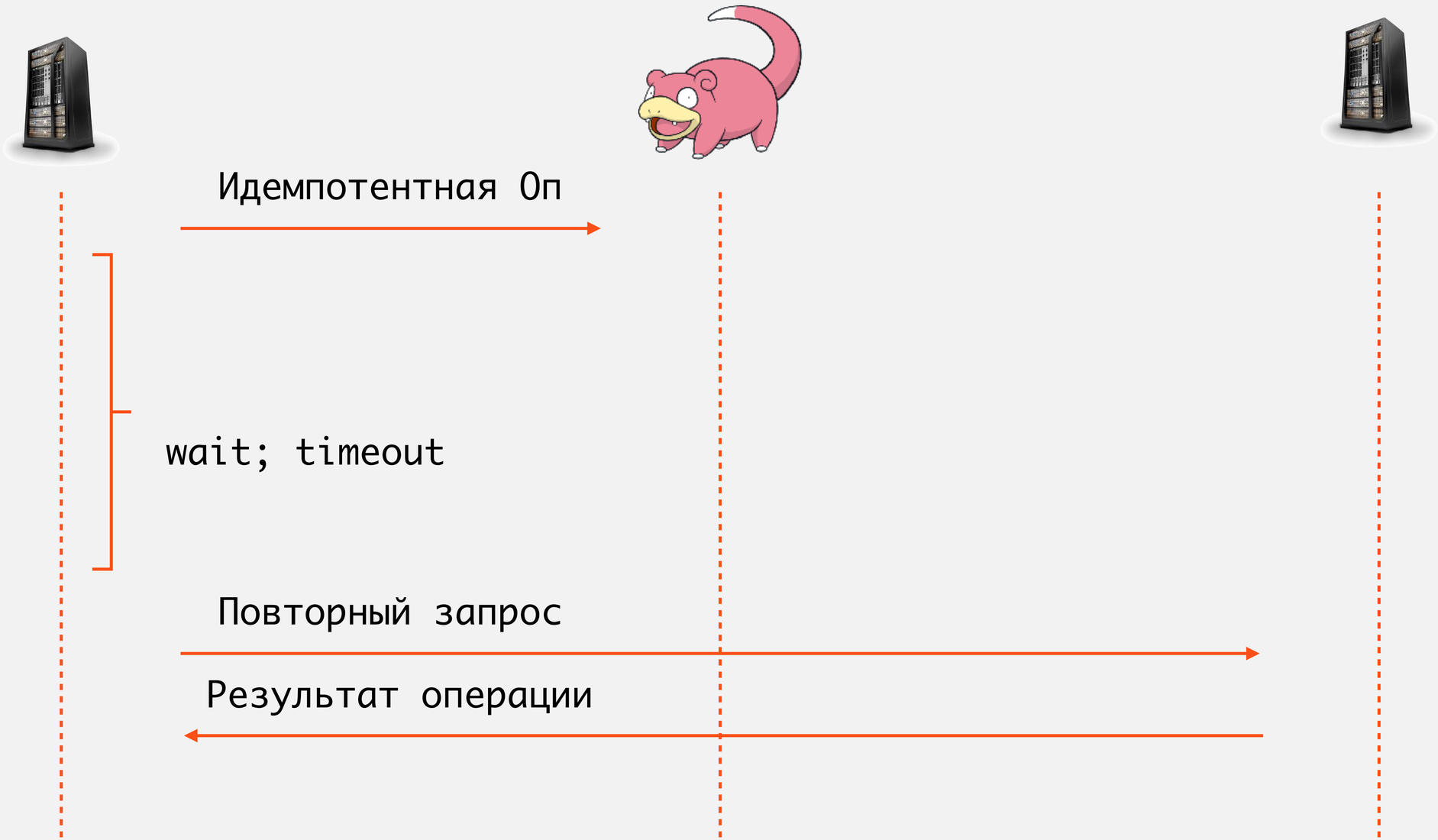
Смерть через торможение



Ставить таймаут = 2.4ms ?

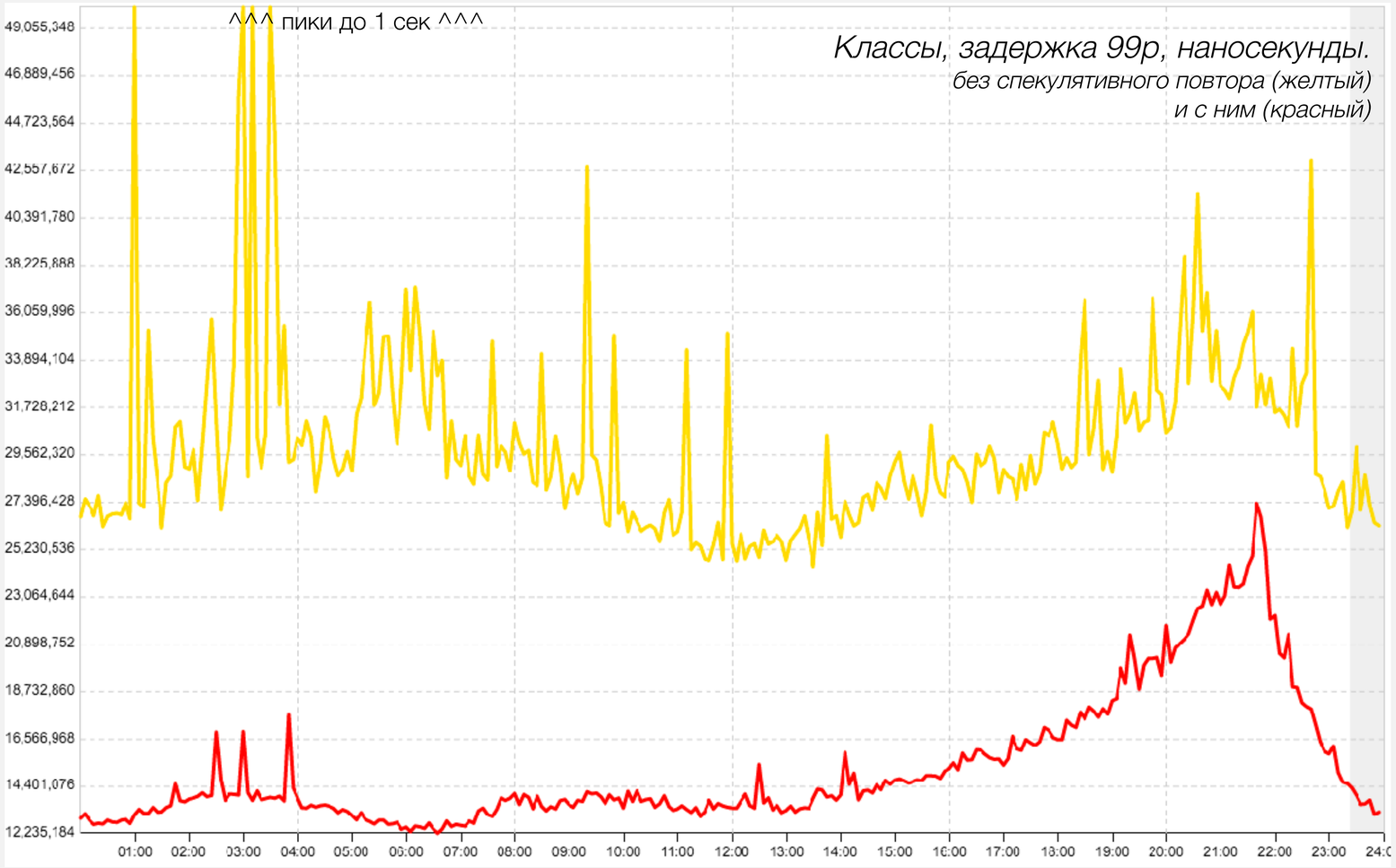
Выводить из ротации если среднее > 2.4ms ?

Спекулятивный повтор



Спекулятивный повтор: что лучше

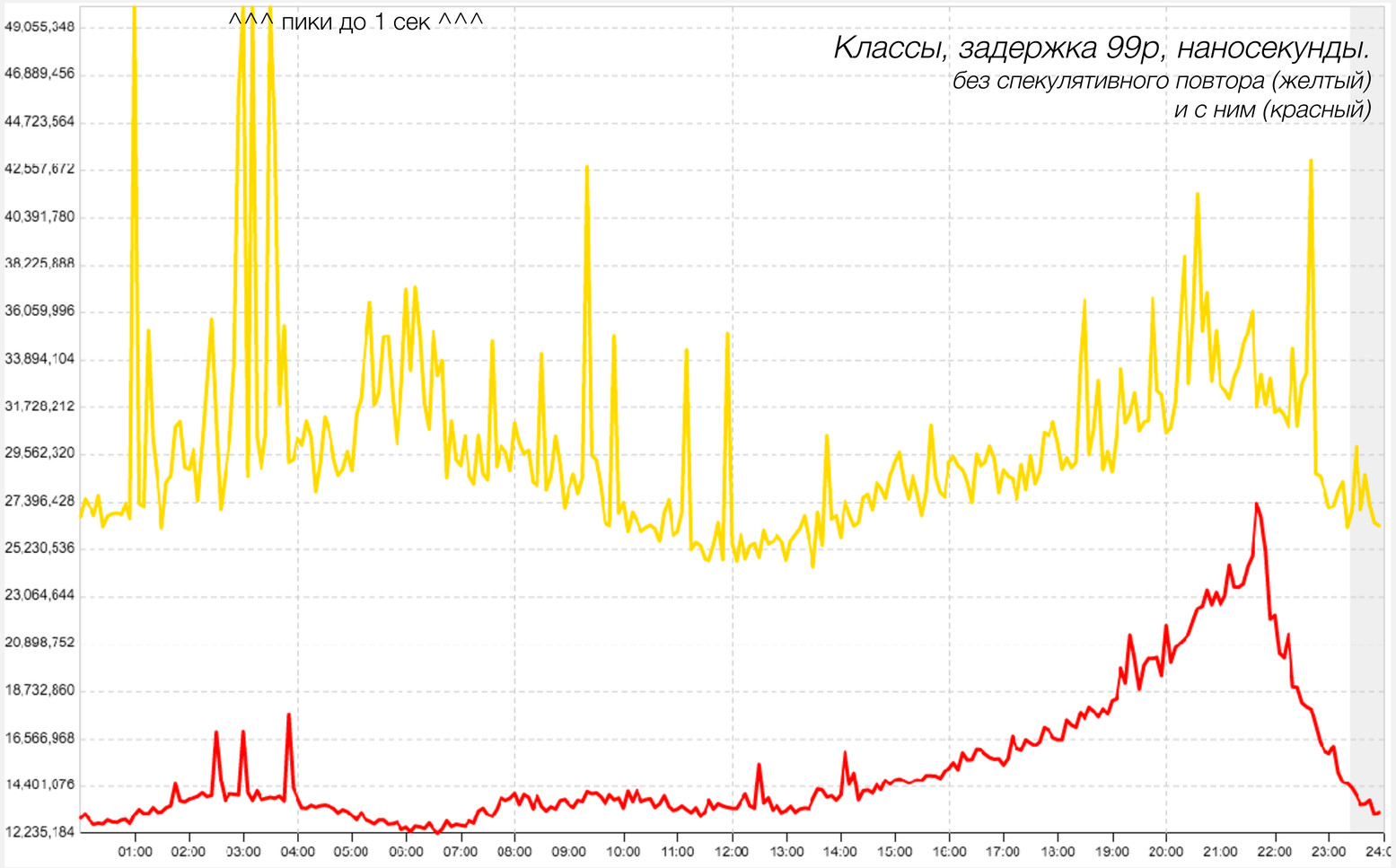
- Задержки 99p, средние
- Стабильность системы



Спекулятивный повтор: применим не всегда

- Идемпотентные операции
- “Дополнительная” нагрузка
- Дополнительный трафик

- Балансируем спекуляцию:
  - всегда, >99p, >50p

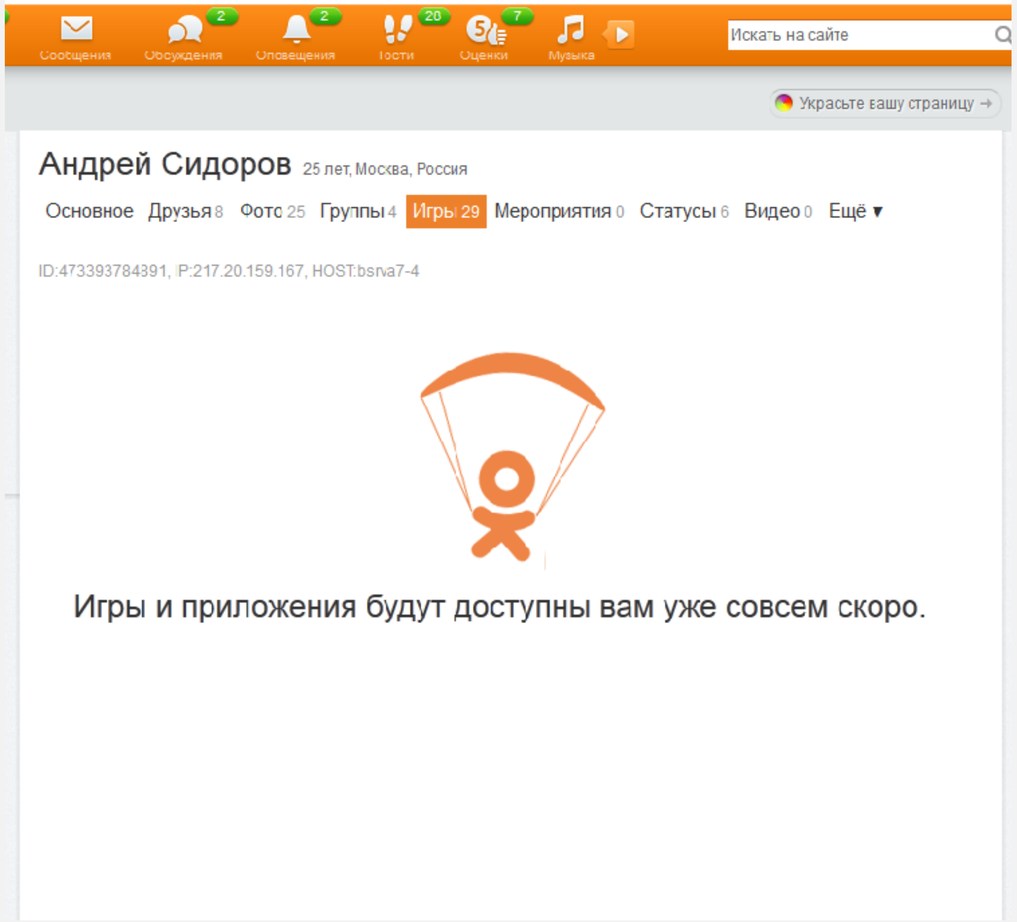


Больше отказов !

Отказ всех реплик сервиса

- Чрезмерная нагрузка
- Чрезмерная паранойя
- Баги
- Люди
- Масштабные аварии

Деградировать!



Использовать другие источники, деградация согласованности

Использовать неполные данные, частичная деградация функции

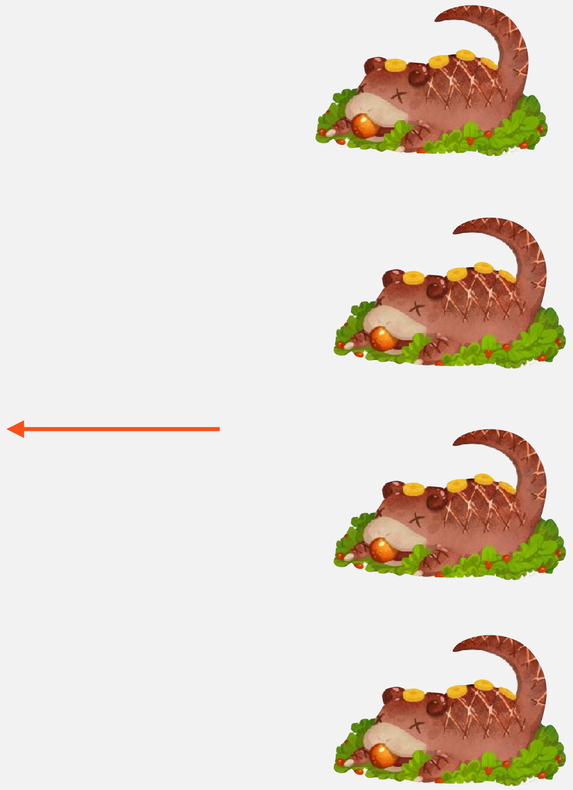
Отключать функцию полностью

Главная СУБД

- Чрезмерная нагрузка
- Чрезмерная паранойя
- Баги
- Люди
- Масштабные аварии

Главная СУБД

- Чрезмерная нагрузка
- Чрезмерная паранойя
- Баги
- Люди
- Масштабные аварии

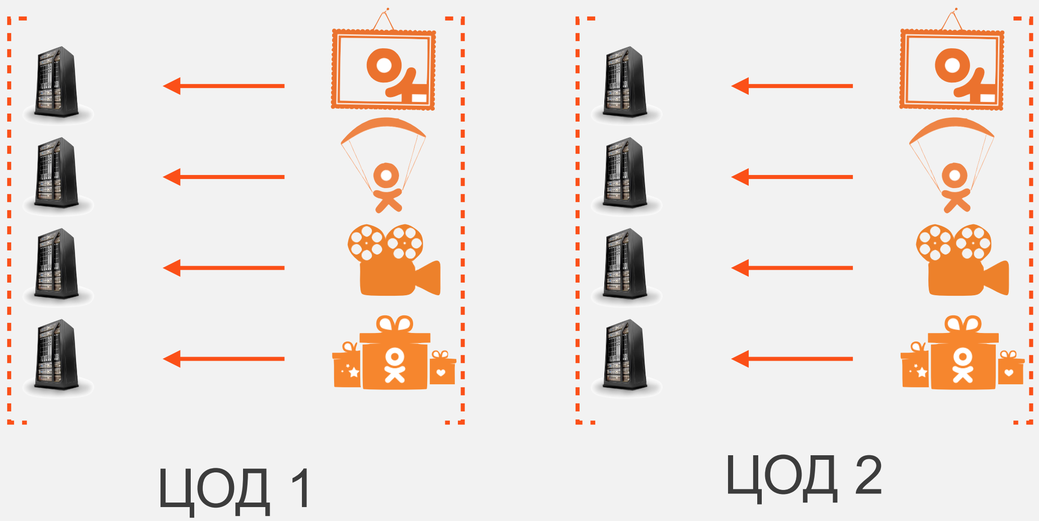


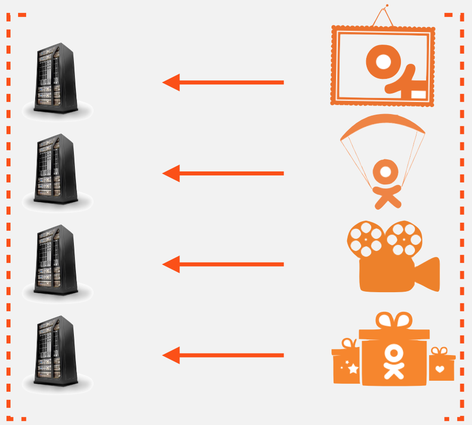
Изолировать !

- Каждому - своя БД
  - или несколько
- Единообразие
- Автоматизация
- Мониторинг

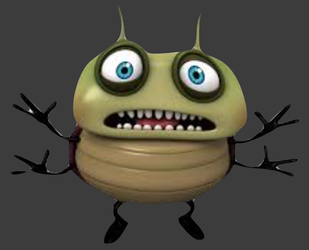
Изолировать !

- 3+ датацентра
- Независимы
- Реплика в каждом
- Работаем с ними по очереди





Релизы



Релизы это:

- Баги
- Десятки фич
- Разные команды
- Частые выкладки
  - Разные платформы
  - AppStore, Google Play

Протестировали

Выложили на препрод

Выложили на прод

Релизы это:

- Баги
- Десятки фич
- Разные команды
- Частые выкладки
  - Разные платформы
  - AppStore, Google Play

Как это тестировать ?

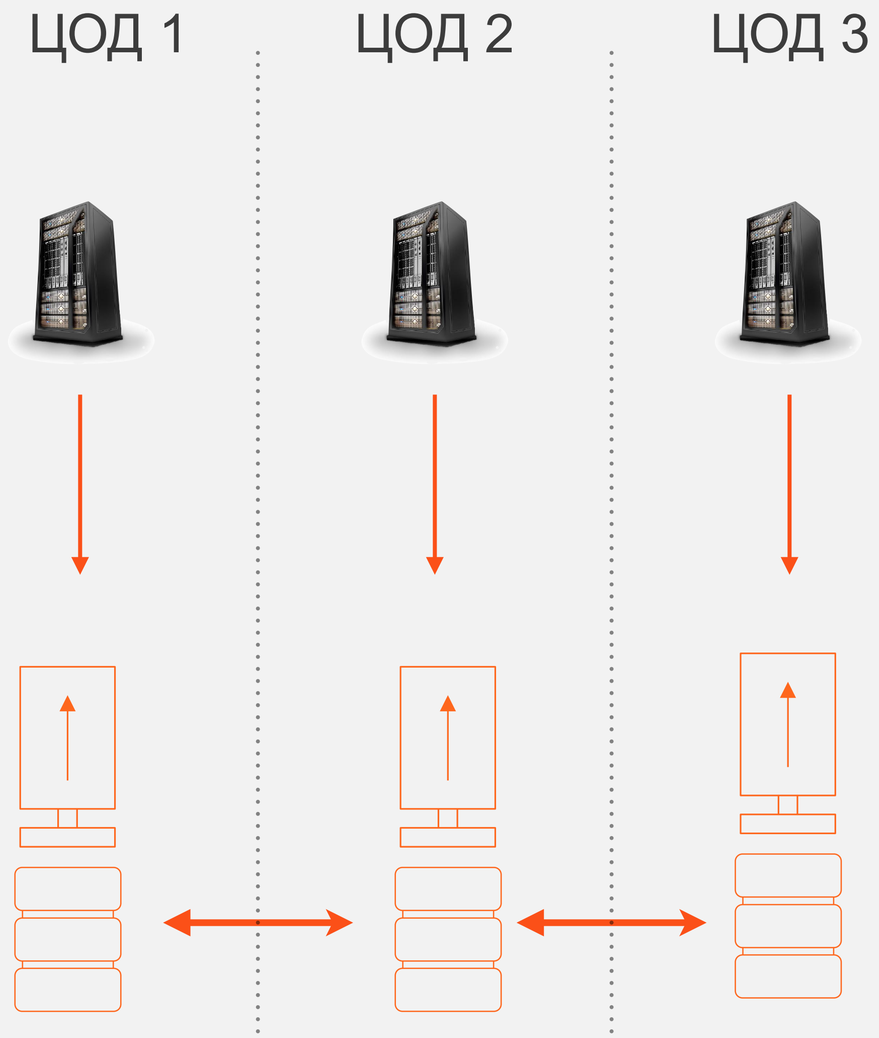
- Тестовый стенд с синтетический нагрузкой ?
  - Другое железо и сеть
  - Сложно воспроизвести профиль нагрузки
  - Для новой фичи редко известен
- На продакшене!
  - Но чтобы никто не заметил

one-conf

- key = value

- Стандартный сервис
- В каждом ЦОД
- Агрессивное кеширование

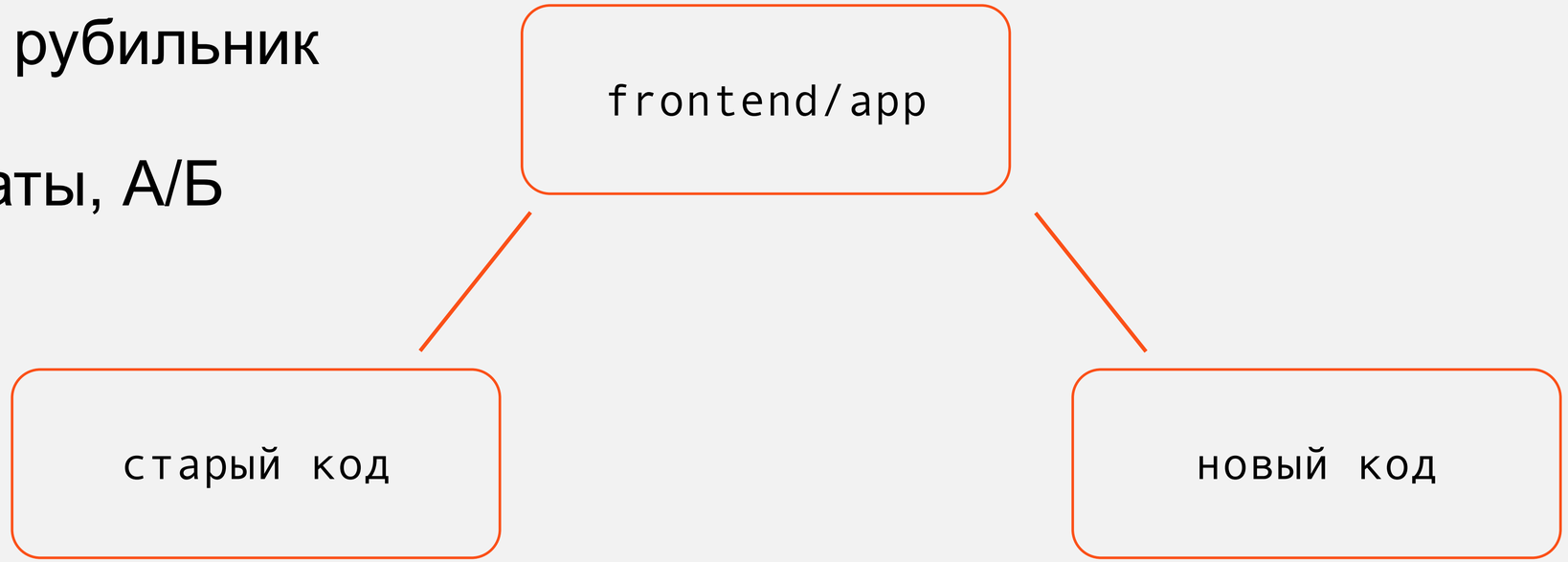
- Рубильники
  - по хосту, группе, ЦОД
  - по гео, партициям, платформам
  - комбинировано
  - нотификация о изменении



Рубильник мудрого джигита

1. Новый код под рубильником
2. Раскатываем с выключенными рубильниками
3. Постепенно включаем рубильник
4. Отслеживаем результаты, А/Б

one-conf

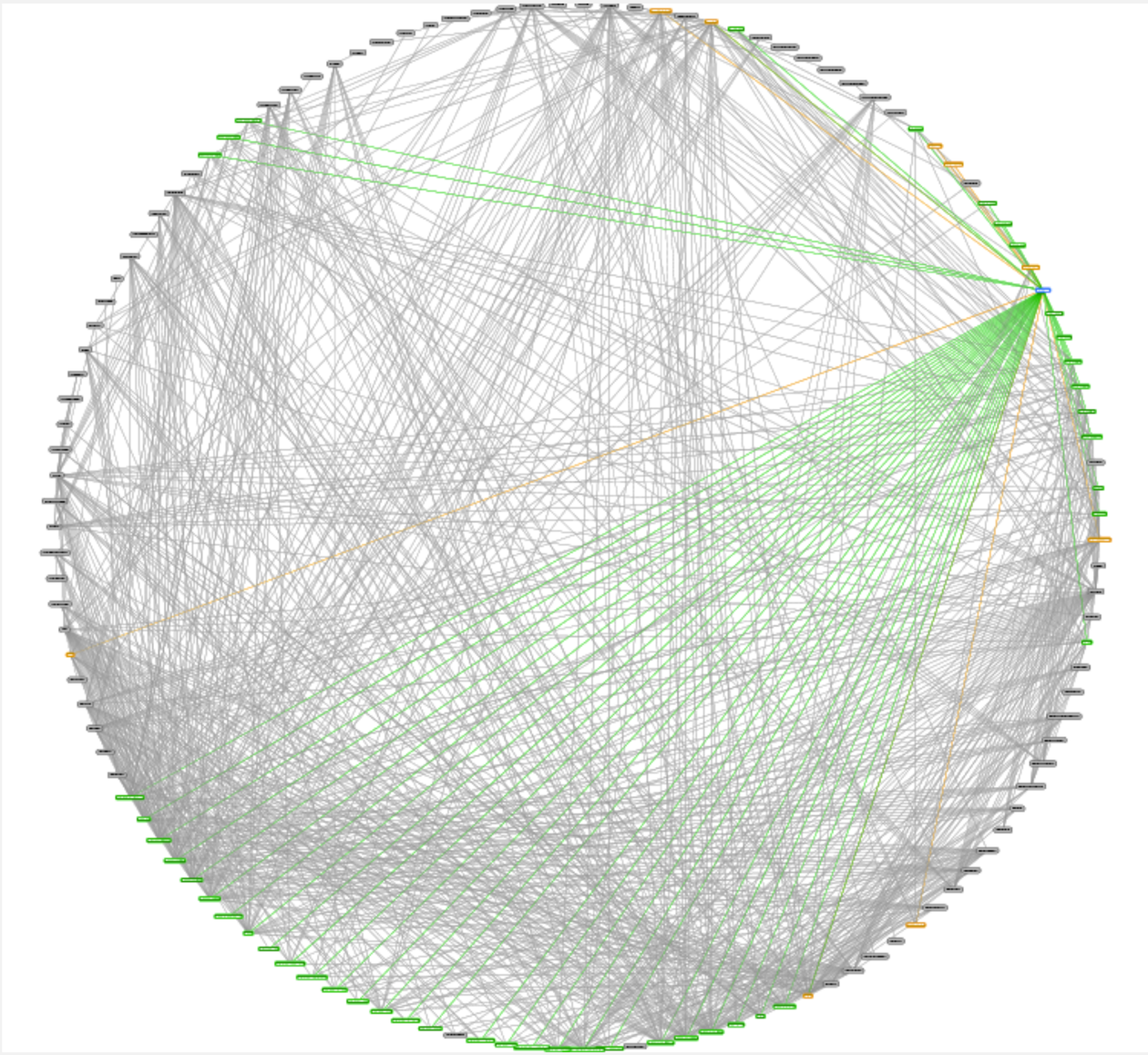


https://en.wikipedia.org/wiki/Feature_toggle

Диагностика

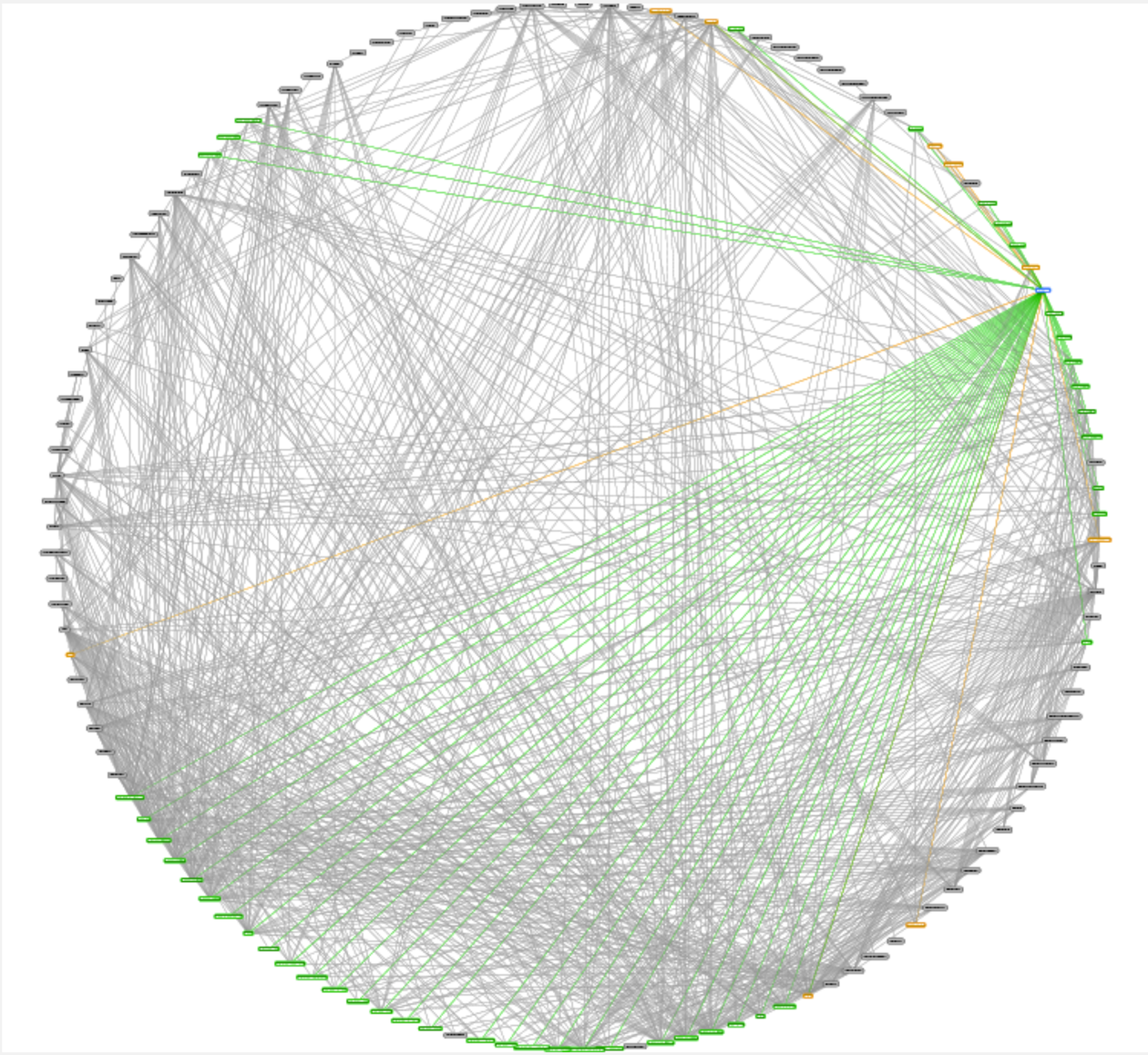
Зачем

- Анализ экспериментов
- Быстрое определение факта аварии
- Локализация проблемы
- Предупреждение аварий



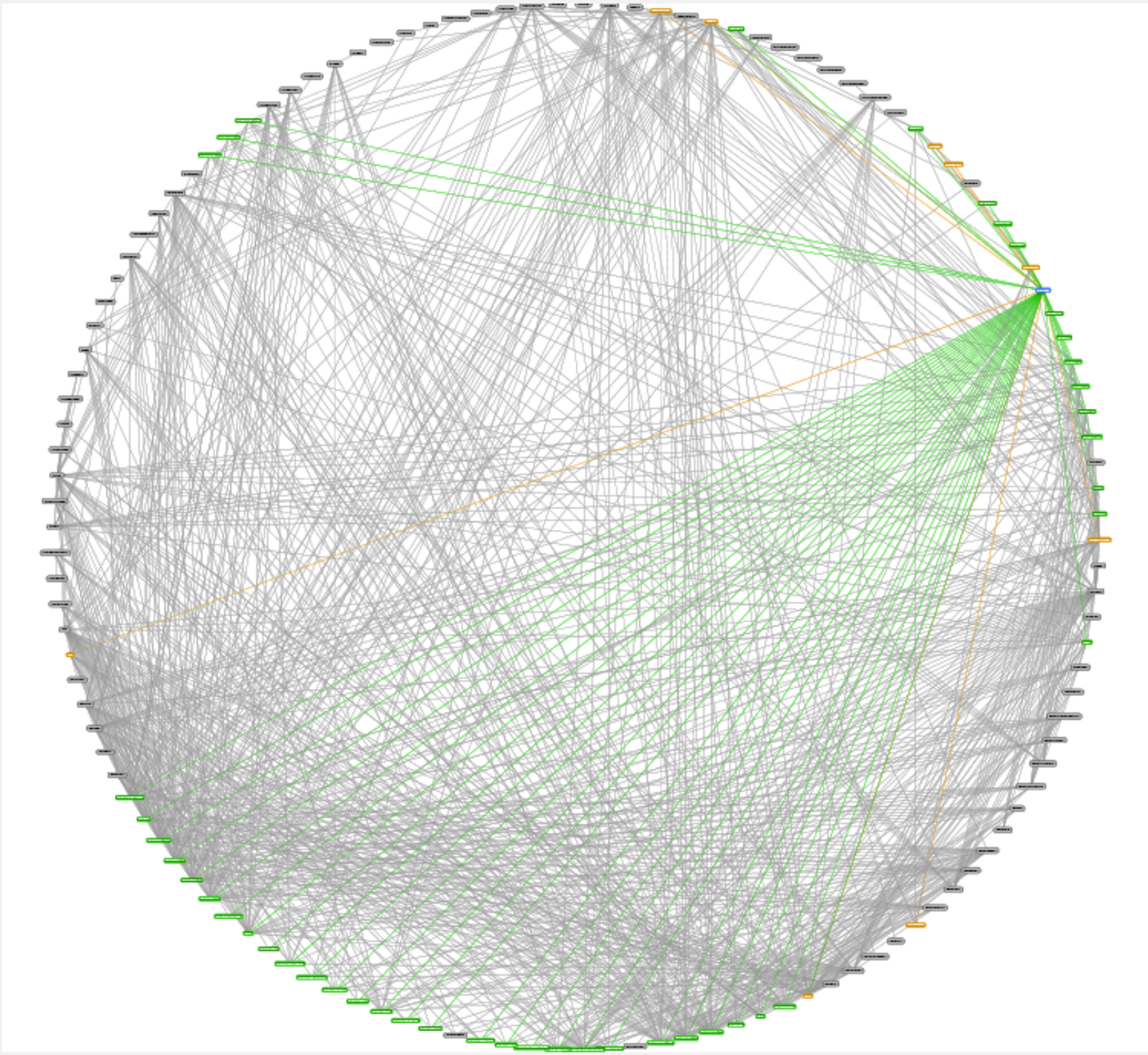
Сбор данных

- Операционные метрики
  - Имена вызванных операций
  - Количество вызовов, успешность
  - Длительность вызовов

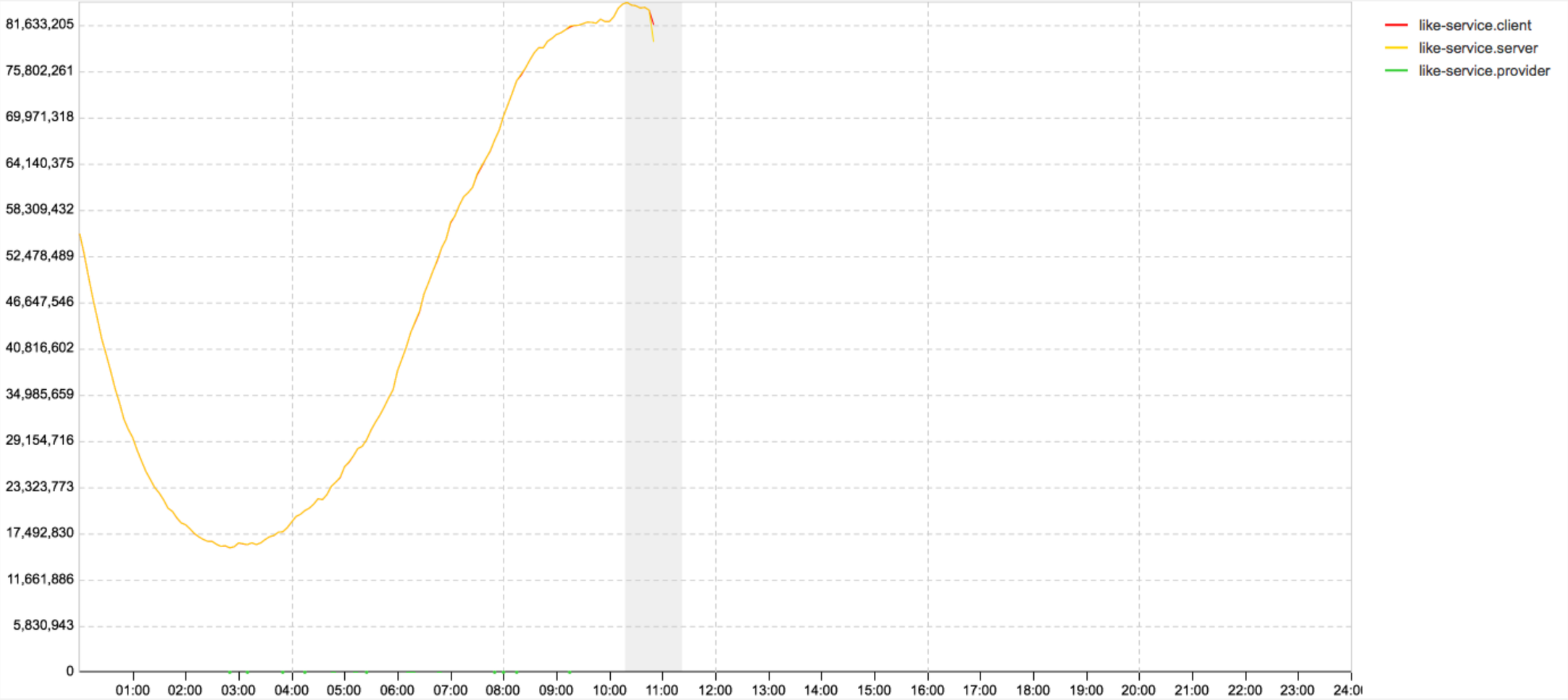


Что показывают графики

- Оперативную статистику и тренды
- Агрегированное число вызовов и ошибок
- Агрегации задержек
  - Среднее, Макс
  - Перцентили 50,75,98,99,99.9

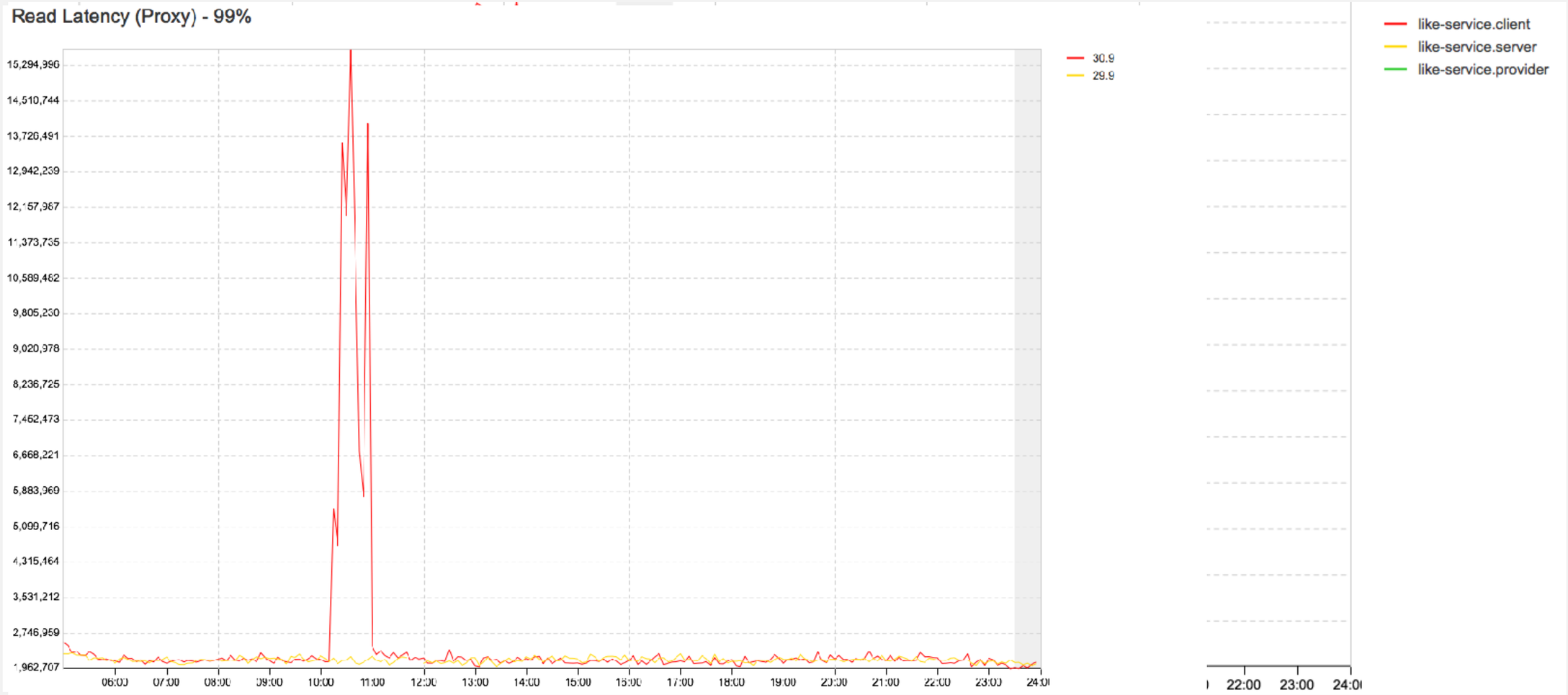


Интересные графики

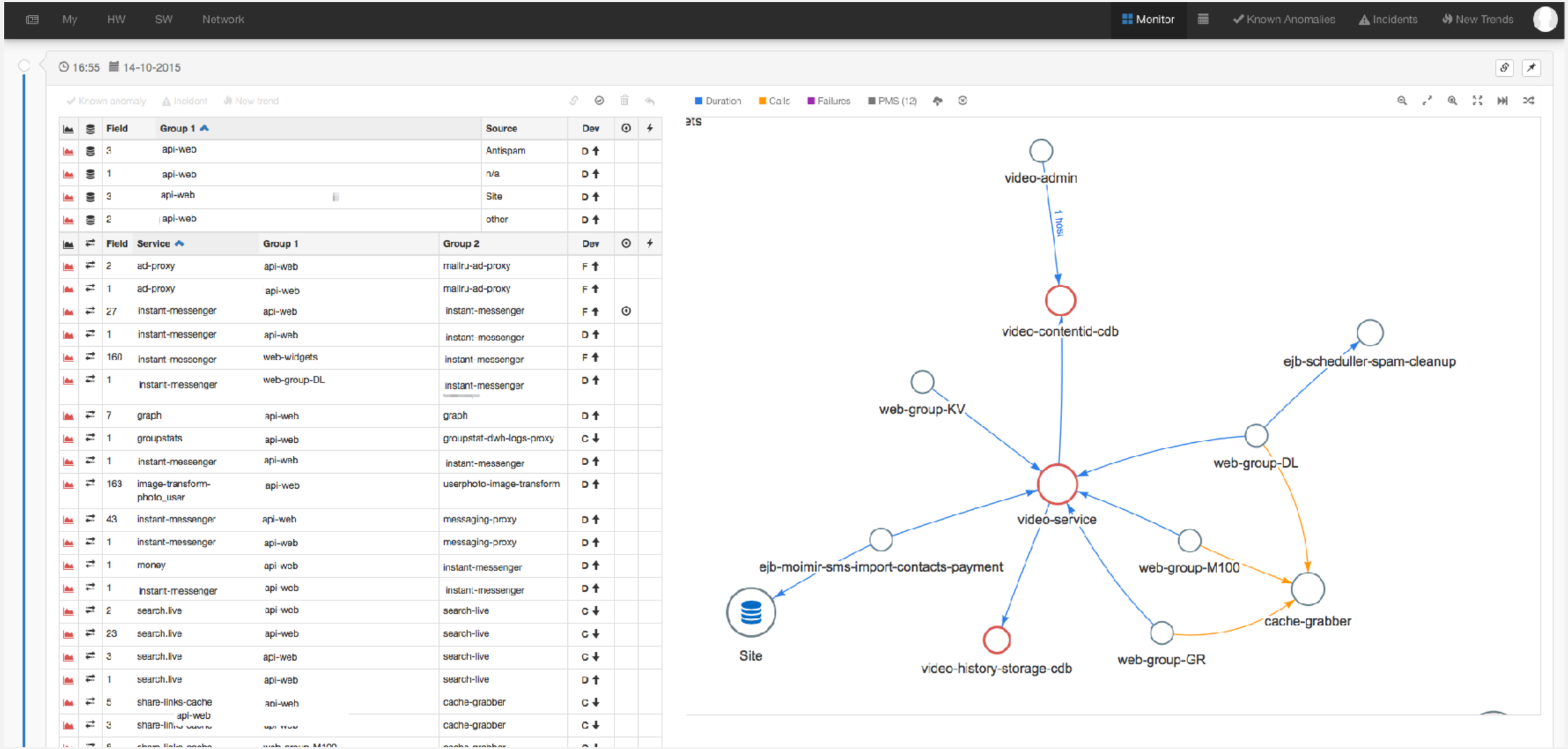


Интересные графики

Интересные графики



Авто поиск аномалий



Field	Group 1	Source	Dev
3	api-web	Antispam	D
1	api-web	n/a	D
3	api-web	Site	D
2	api-web	other	D

Field	Service	Group 1	Group 2	Dev
2	ad-proxy	api-web	mailru-ad-proxy	F
1	ad-proxy	api-web	mailru-ad-proxy	F
27	instant-messenger	api-web	instant-messenger	F
1	instant-messenger	api-web	instant-messenger	D
160	instant-messenger	web-widgets	instant-messenger	F
1	instant-messenger	web-group-DL	instant-messenger	D
7	graph	api-web	graph	D
1	groupstats	api-web	groupstat-dwh-logs-proxy	C
1	instant-messenger	api-web	instant-messenger	D
163	image-transform-photo_user	api-web	userphoto-image-transform	D
43	instant-messenger	api-web	messaging-proxy	D
1	instant-messenger	api-web	messaging-proxy	D
1	money	api-web	instant-messenger	D
1	instant-messenger	api-web	instant-messenger	D
2	search.live	api-web	search-live	C
23	search.live	api-web	search-live	C
3	search.live	api-web	search-live	C
1	search.live	api-web	search-live	D
5	share-links-cache	api-web	cache-grabber	C
3	share-links-cache	api-web	cache-grabber	C

Краткое содержание предыдущих слайдов

- Возможности отказов в распределенных системах безграничны
- Отказы маскируются за счет информации, времени, железа
- При немаскируемых отказах — деградируем !
- Изоляция отказов - важная часть надежности
- Отказы важно диагностировать и предупреждать на проде

Тут можно узнать больше:

https://v.ok.ru/publishing.html

Лекции Технополиса.
Проектирование высоконагруженных систем
упороться на 30 часов тут:

https://habr.com/company/odnoklassniki/blog/347798/