Как мы переносим миллионы пользователей между дата-центрами

Крашенинников Александр
Badoo

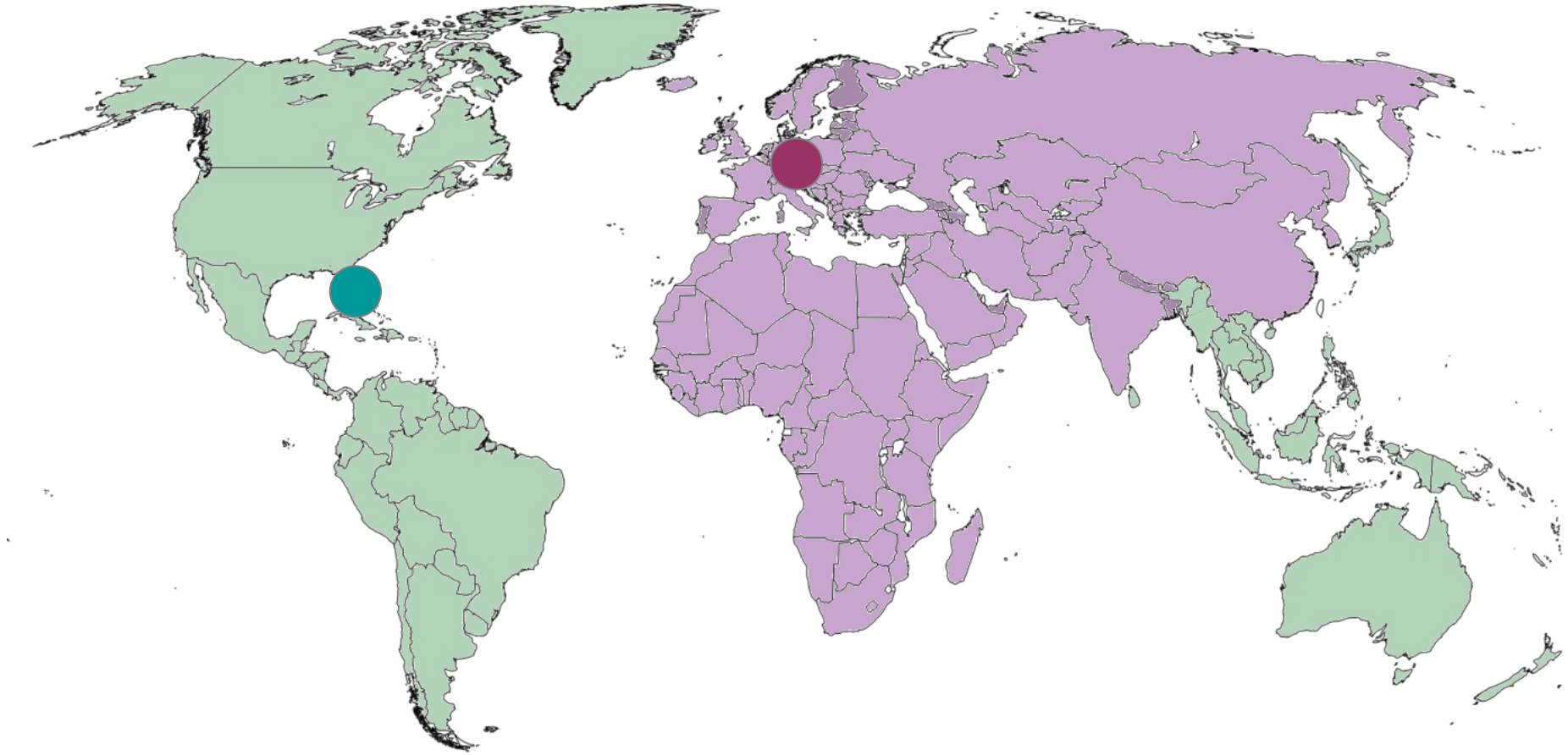
Про что это, вообще

- Зачем переносить данные
- Архитектура нашего решения
- Жизнеобеспечение процесса

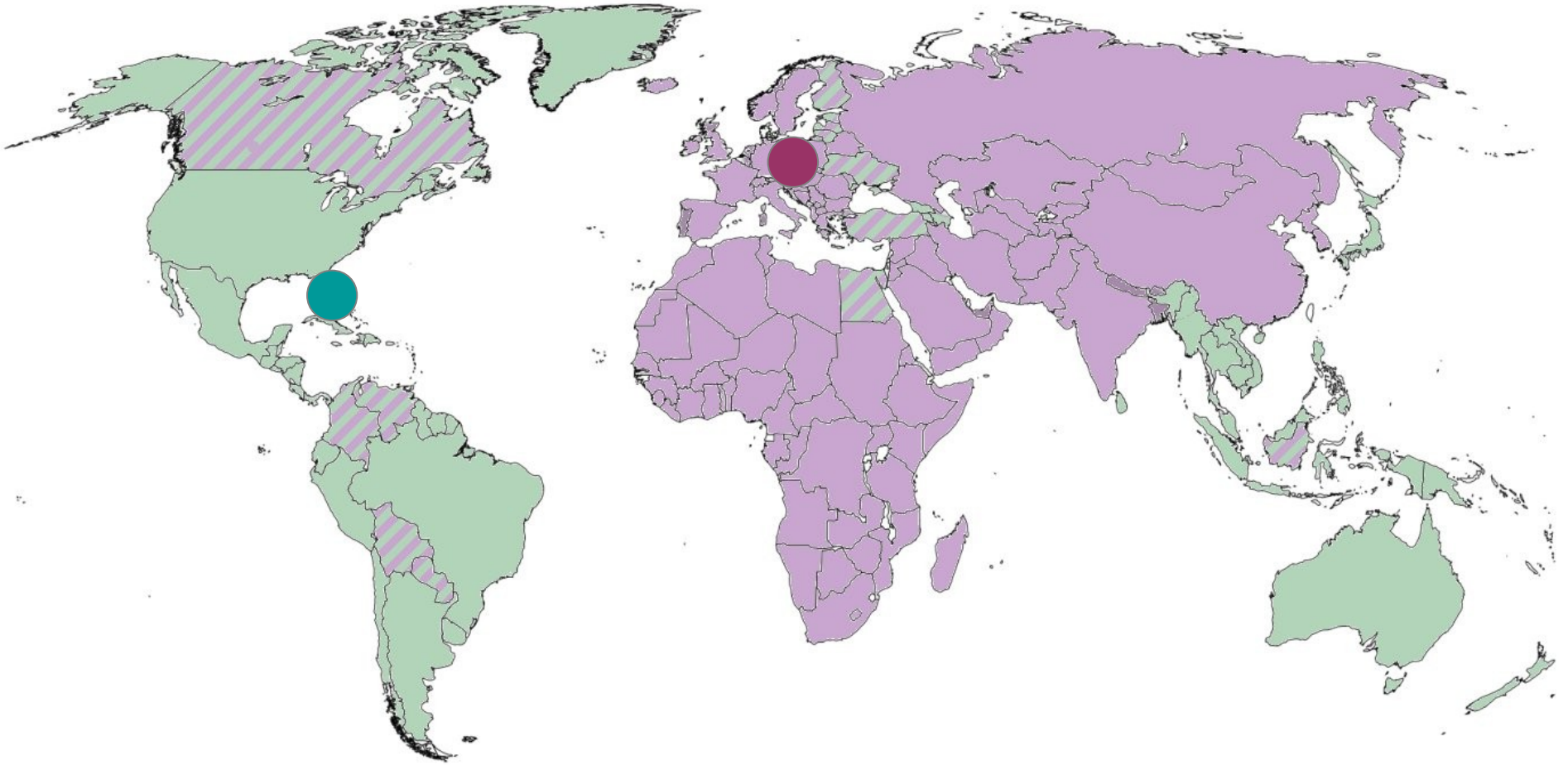
Второй слайд всех презентаций Badoo

- >250M пользователей
- 1Pb фотографий и видео
- 2.5 датацентра
  - Прага
  - Майами
  - Гонконг (CDN)

География обычных пользователей



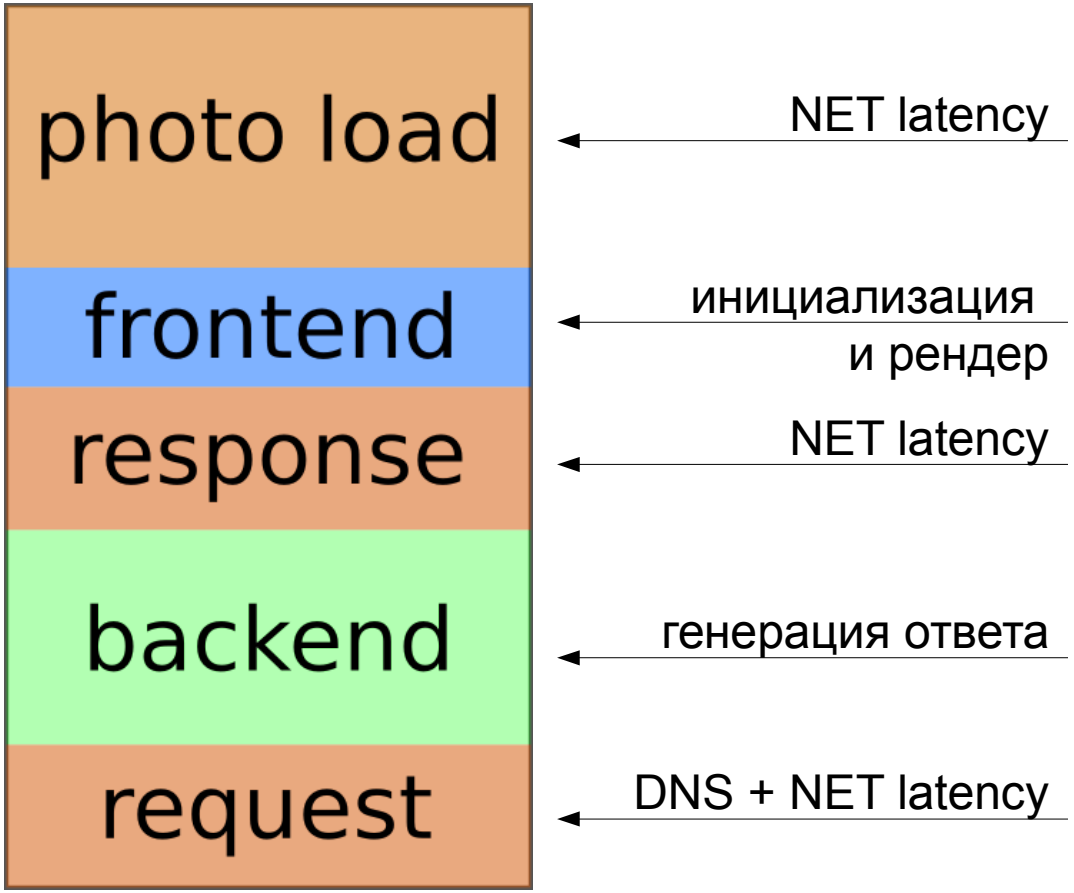
География курильщиков



Как понять, что надо переезжать?

Ответ: Время отклика сайта

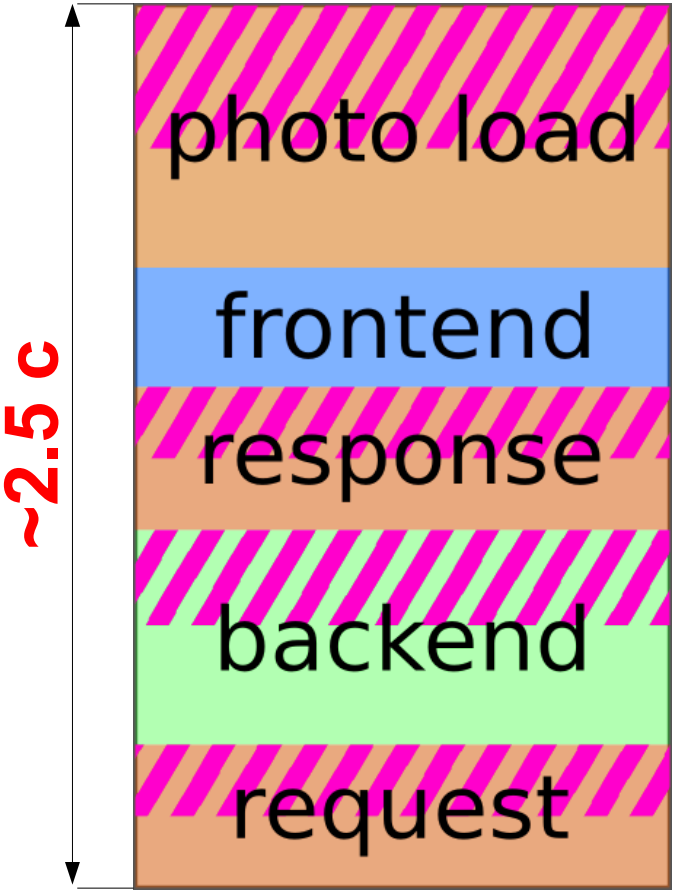
UX - кто он?



Измерено с помощью Jinba

http://tiny.cc/jinba-slides
http://tiny.cc/jinba

UX - есть что улучшить!



накладные расходы
на взаимодействие
с другим ДЦ

Хотелки

- Уменьшить время на взаимодействие
- Максимально незаметно для пользователя

Вопросы по хотелкам

Кого? Что? Как?

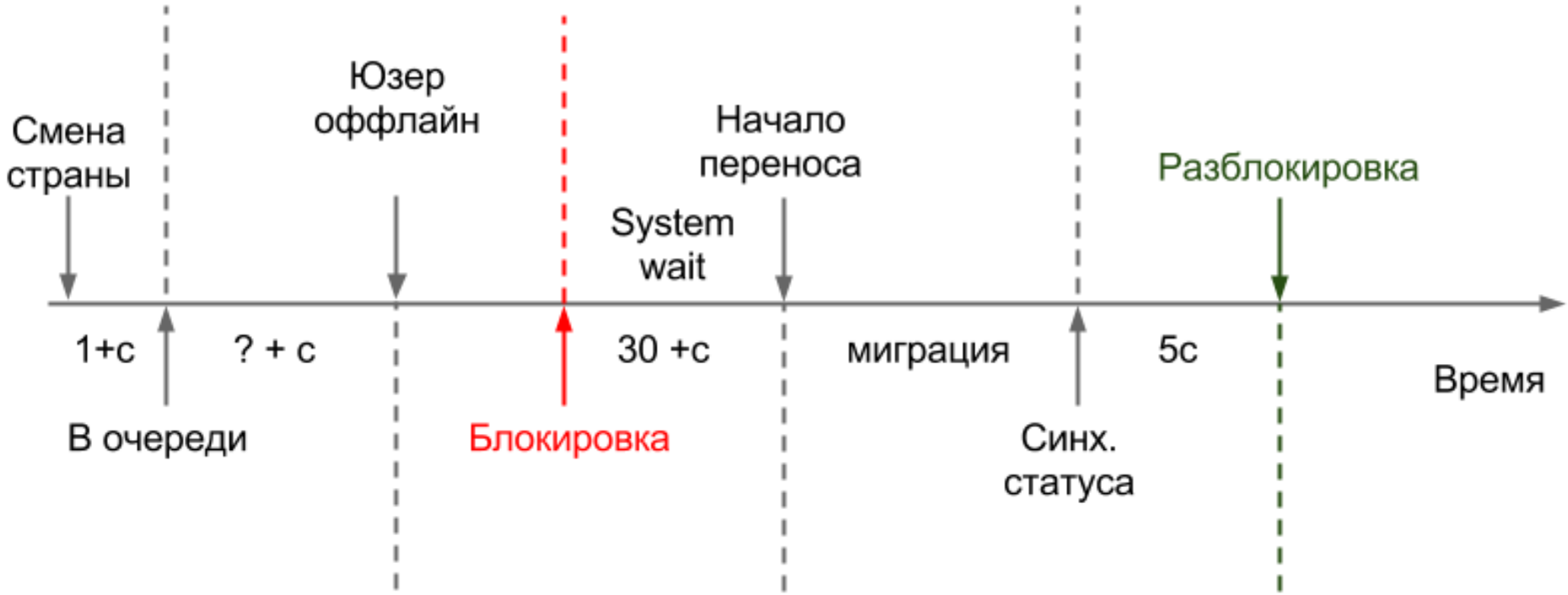
…

переносить

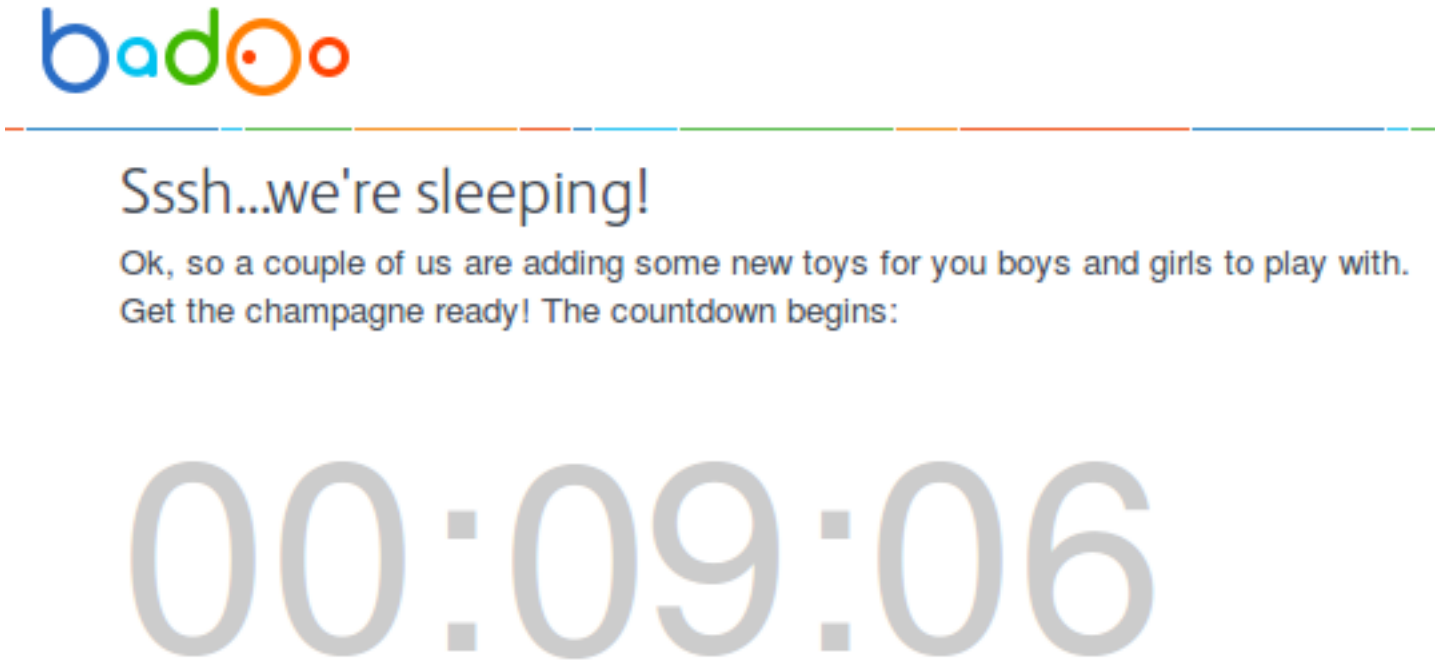
Кто переезжает

- Сменившие страну на принадлежащую другому ДЦ
- «Исторически» не там (8M пользователей на оба ДЦ)
- Плановый перенос азиатских стран (20M пользователей)

Как происходит переезд



Как выглядит для пользователя



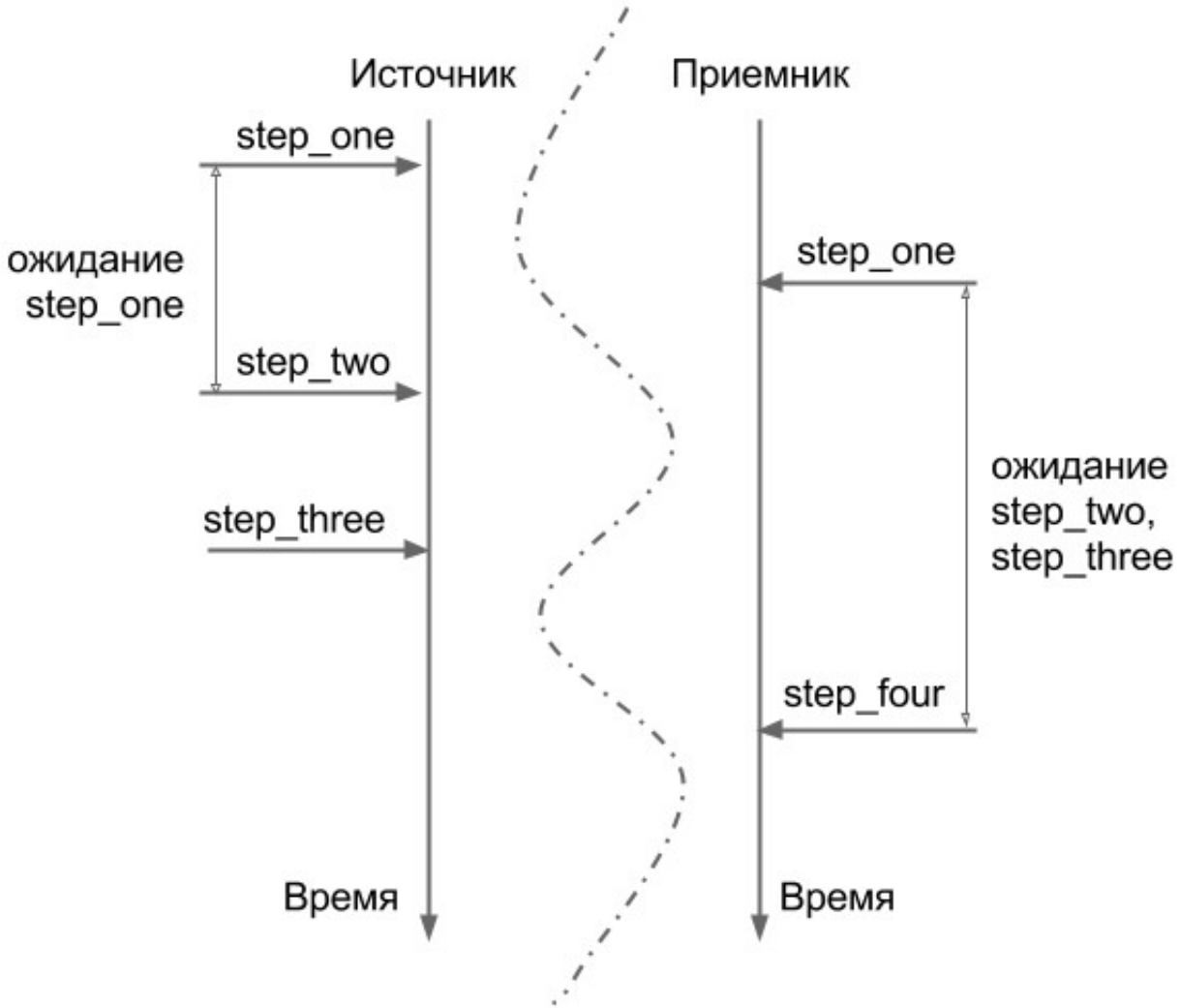
Процесс переноса = цепочка шагов

- Для каждого шага характерны:
  - Целостность (!!!)
  - Идемпотентность
  - Отсутствие сайд-эффектов

Шаги - конфигурация

- Где выполнить
  - Каждый ДЦ (источник и приемник)
  - Один из двух
- Последовательность
- Пауза
  - Отложить обработку пользователя

Цикл миграции - последовательность шагов



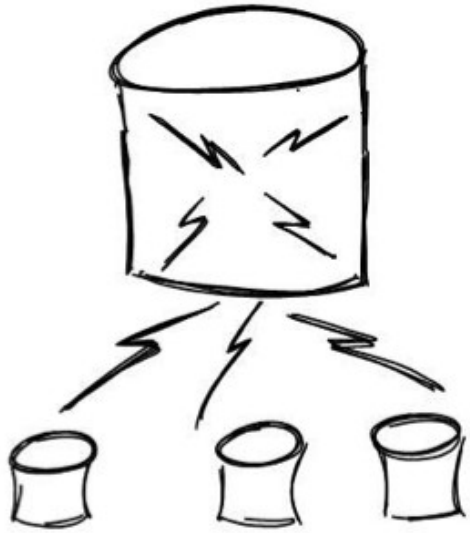
Цикл миграции - хранение состояния

user_id	shard_id	step_one	step_N	delayed	delete	errors
10	1	1,2		2015-10-10	0	15
20	2	1		NULL	0	0

- На каждое направление миграции – таблица
- Денормализованное хранение исполненых шагов
-
```
UPDATE #TABLE# SET step_#step# =
IF(step_#step# = '', '#dc_id#',
CONCAT(step_#step#, ',#dc_id#'))
WHERE user_id = #user_id#
```

Из чего сделан юзер - User DB

- Каждому юзеру — db_shard
- db_shard = (dbN.host,dbX_index,tableY_index)
- Каждый шард реплицируется в другой ДЦ
- Тысячи их!



Миграция базы

- Выбрать данные из источника
- Модифицировать (опционально)
- Сделать дамп с транзакцией
- Применить дамп
- Удалить в источнике

Миграция базы - выборка

1. Получить список таблиц на хосте
2. BEGIN TRANSACTION;
3. SELECT * FROM Messages.Incoming WHERE $COND1;
4. SELECT * FROM Votes.UserVote WHERE $COND2;
5. SELECT * FROM DbN.TableY WHERE $COND3;
6. COMMIT;

Миграция базы - подготовка дампа

1. Замена данных, контекстных ДЦ-источнику
2. Запись в файл
    1. BEGIN TRANSACTION;
    2. Данные из источника
    3. COMMIT;
3. Проверить что все данные корректно записаны!

Миграция базы - применение дампа

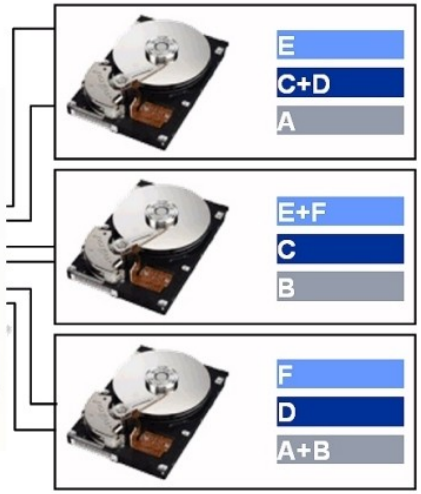
1. ssh $destination
2. В STDIN скармливаем файл с дампом
3. Пишем в файл $destination:/$file.sql
4. Проверяем MD5 (!!!)
5. Mysql < /$file.sql; rm /$file.sql
6. На источнике – сохранение файла в бекап

Миграция базы - удаление в источнике

1. Из тех же таблиц, откуда выбирали
2. Обязательно отдельным шагом

Из чего сделан юзер - фото и видео

- По аналогии с базой — photo_shard
- photo_shard = (photoN.host,/disk/path/to_shard)
- Реплики в другой ДЦ нет
- Plain files (no BLOB)



Перенос фотографий

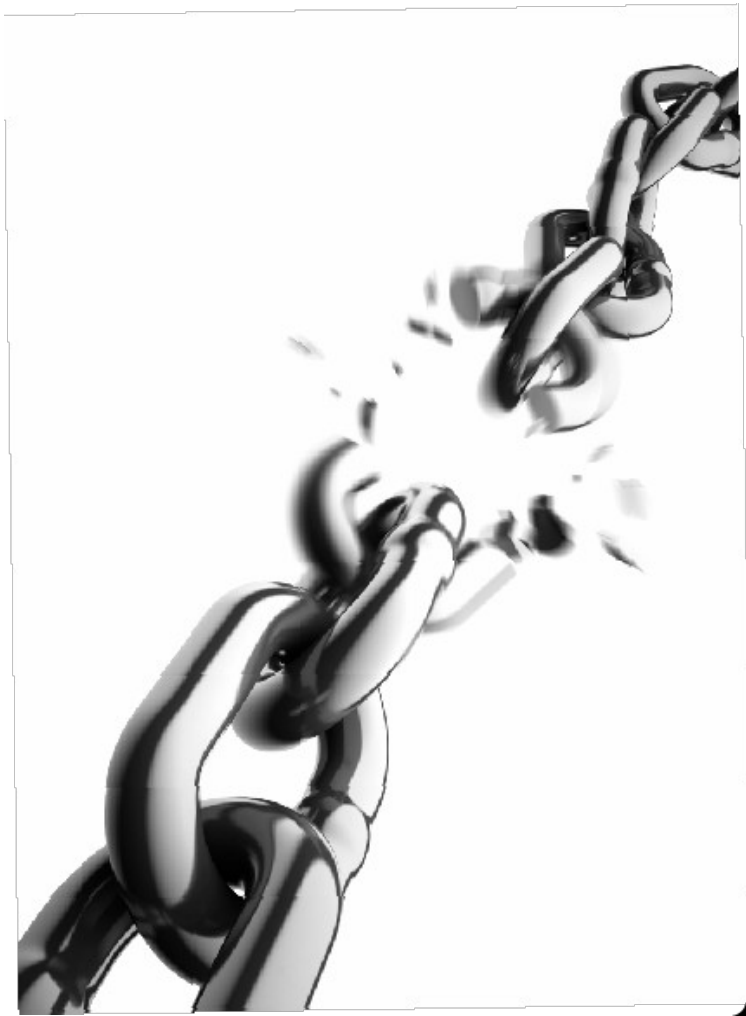
- Проверка источника и приемника
- Rsync файлов
- Задание на backup
- Обновление ссылок на photo_shard

Обновление сервисов

- Синхронный RPC-вызов
  - Update
  - Delete на источнике + insert на приемнике
- Запись в БД
  - Аналогично миграции

Ошибочка вышла!

- DB: дамп не применился
- Фотки: rsync сломался
- Сервис: недоступен



Миграция очередей

- Происходит в момент обработки события
- Обработчики пропускают события при миграции
- После переезда пользователя, события переносятся (1 Thread)

MigrationAPI - мигрирует ли <user_id>

- Файл на локальной FS
- Нет в файле → не мигрирует 100%

1. Код смотрит:
   а.) Файл
      б.) Memcache
         в.) MySQL

- Генерируется на 2 машинах + curl с каждого хоста (~800 в каждом ДЦ)

Окружение

- «Облачный» запуск PHP-скриптов миграции
- Микрооблако из 4 машин
- (400 источник + 400 приемник) x 2 ДЦ = 1600 процессов
- MySQL master-master между ДЦ

Параллелизм обработки

- Один процесс работает с одной user DB
- Процесс имеет свой номер шарда
- Автоматический ребалансинг на основании статистики

Интеграция

- Обслуживание user DB
  - Альтеры: по расписанию, stop migration
  - Проверка отставания репликации для нового db_shard пользователя
- Maintenance фотосерверов
  - Циклические ошибки:
    - Другой фотосервер
    - Отмена миграции

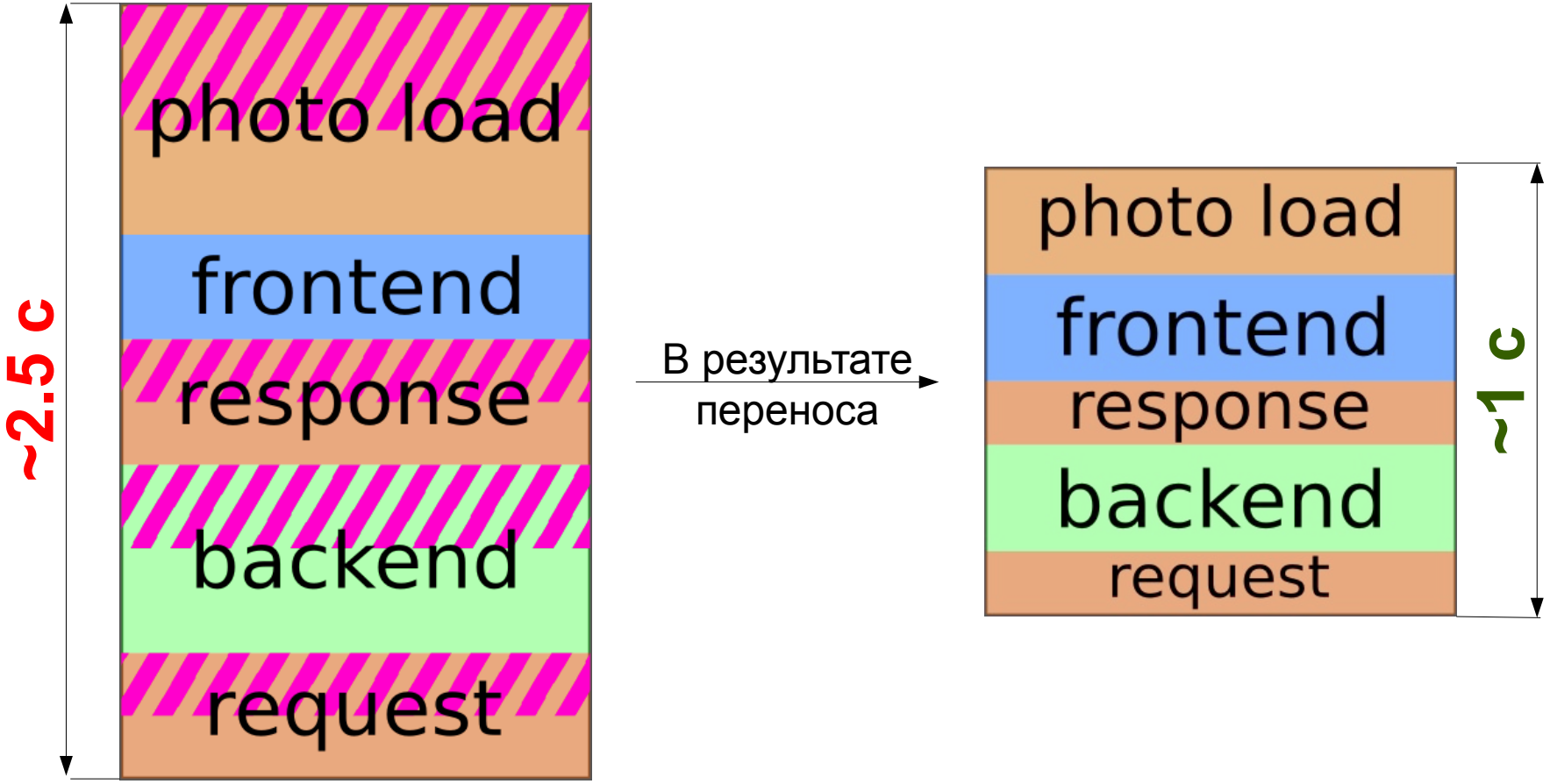
Статистика

- Корректность работы
  - Длина очереди
  - Время недоступности сайта
- Статистика шагов
  - Ошибки
  - Время исполнения

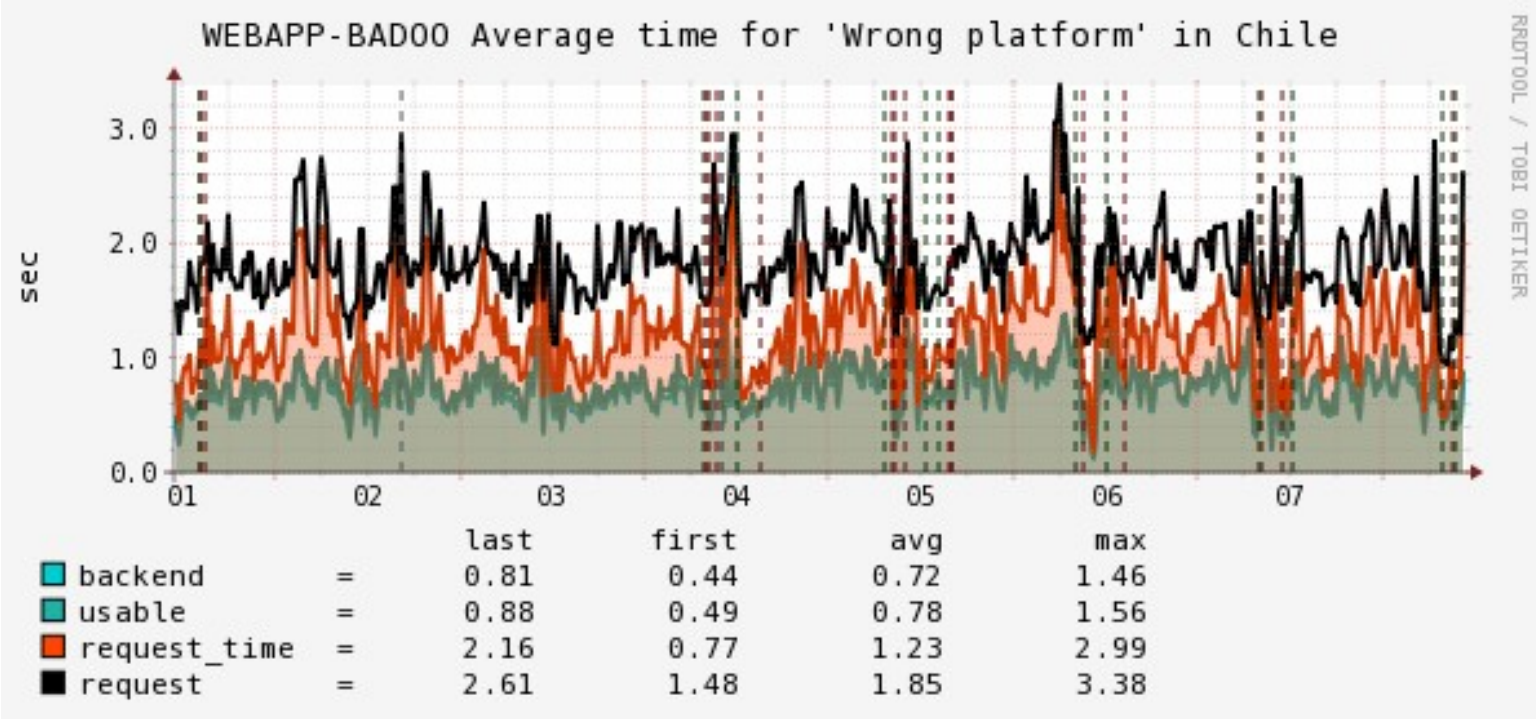
Мониторинг

- Как много пользователей ждут?
- Распределение по времени блокировки
- Абсолютное число ошибок на шагах

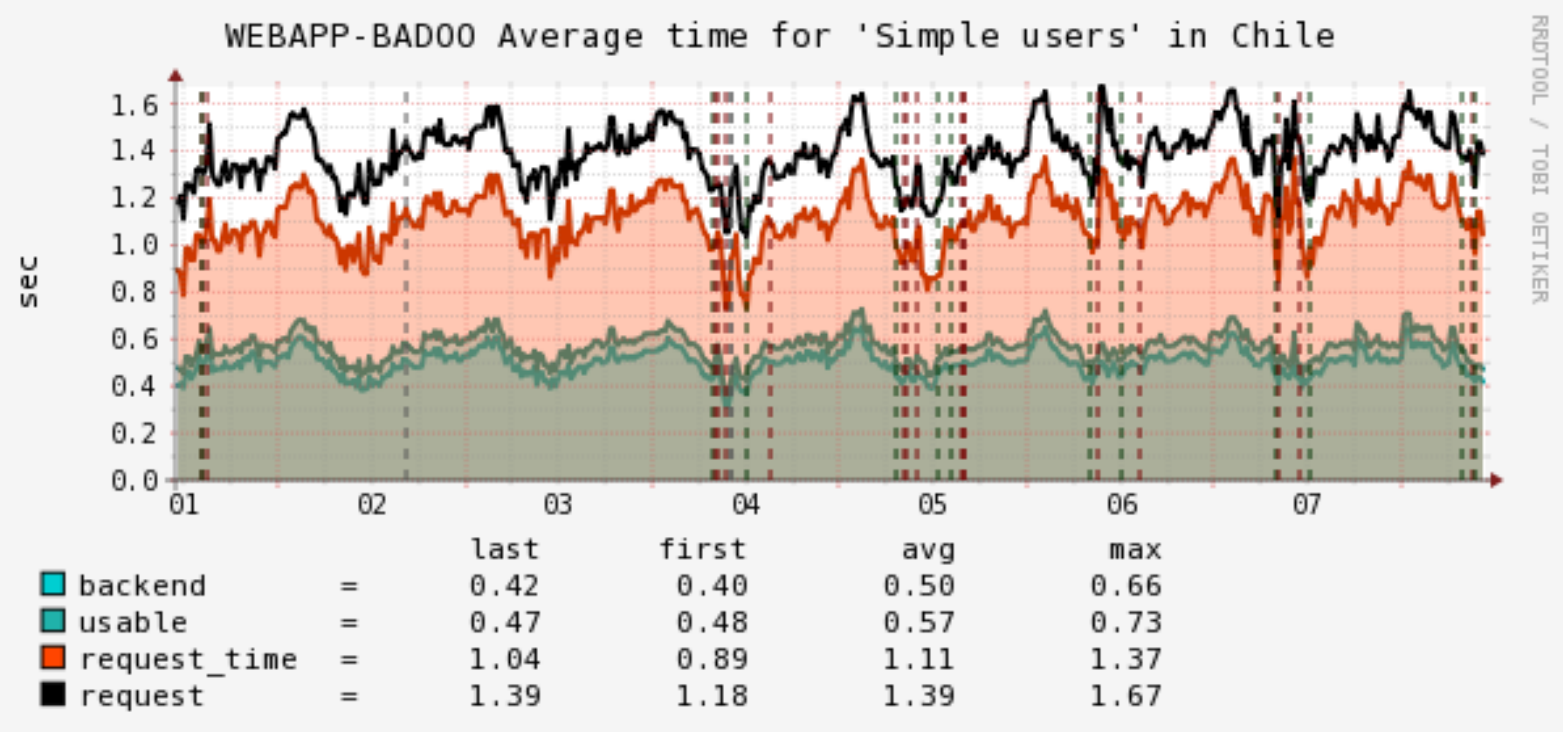
Итоги - улучшен UX!



В Чили всё было плохо



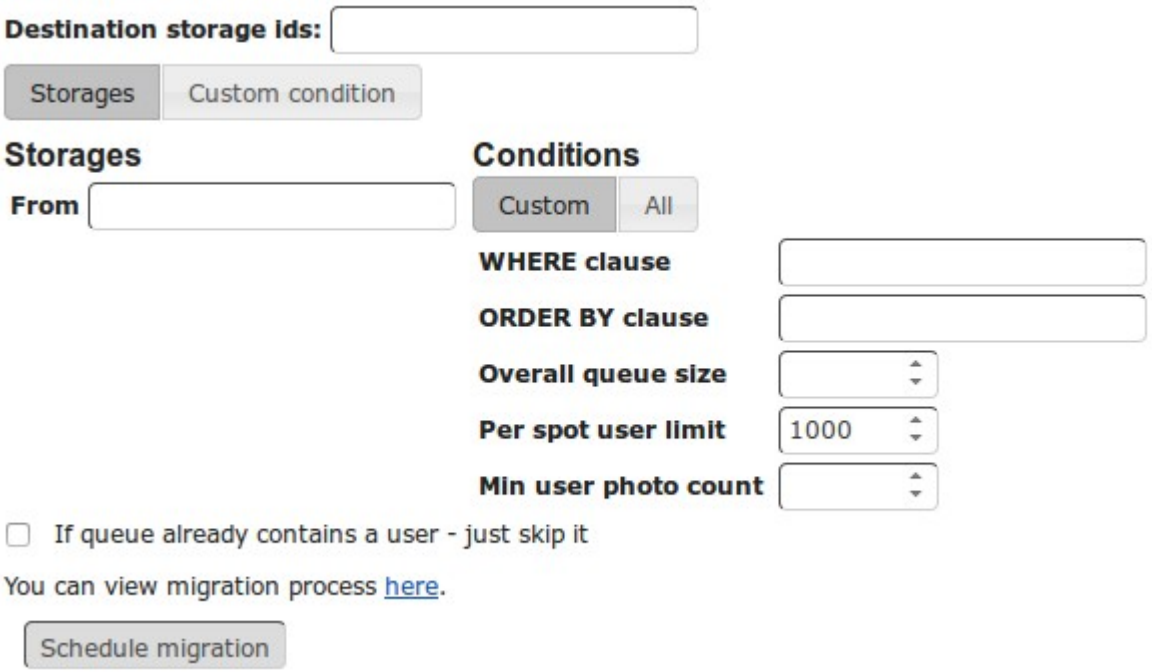
В Чили всё стало хорошо!



Внутренние инструменты

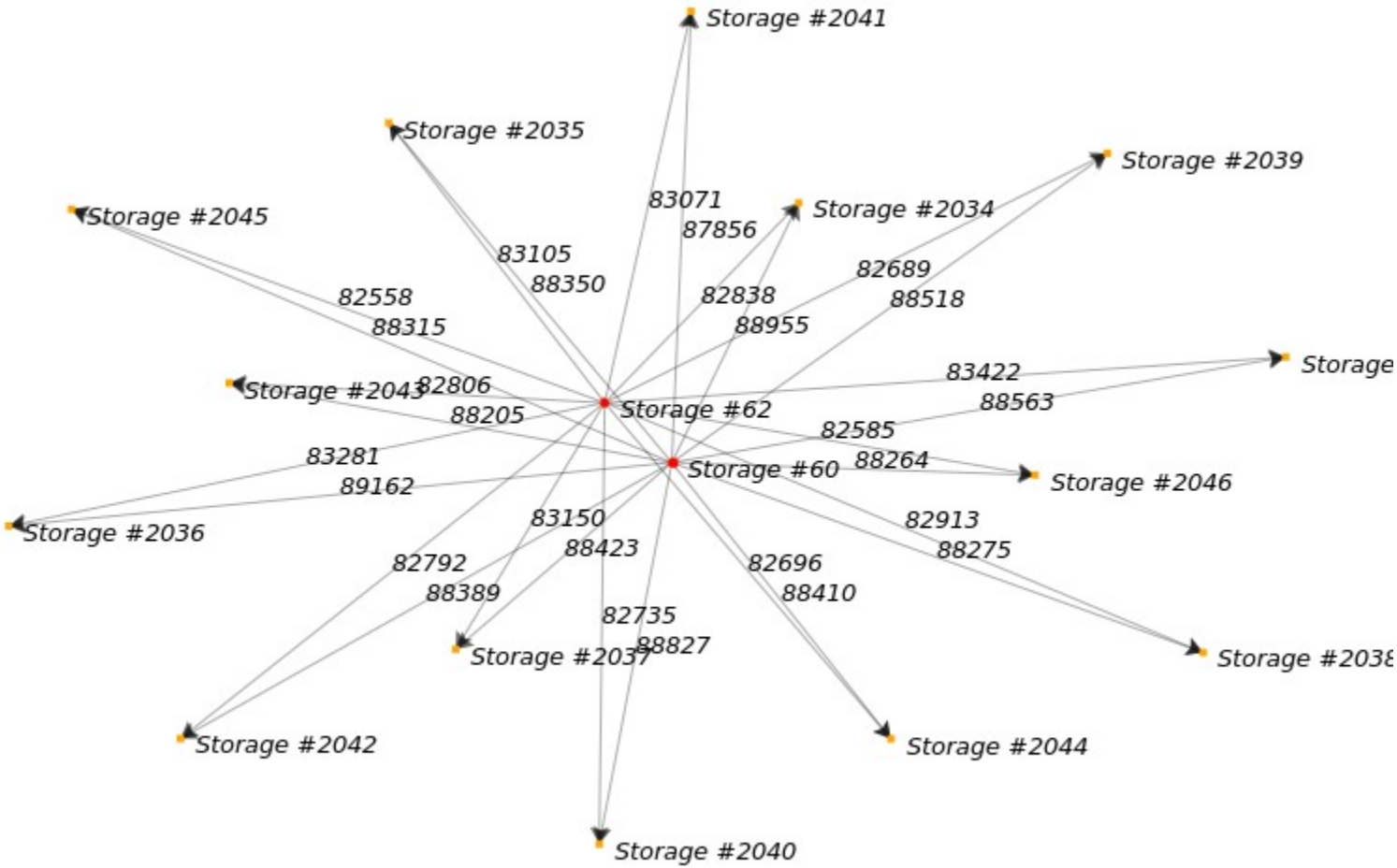
- Управление данными внутри одного ДЦ
  - базы
  - фотки

Миграция фотосерверов



- Заполнить условия для переноса
- Ждать

Миграция фотосерверов



Миграция user DB

Schedule

Source DBS

dbs2.ulan
dbs3.ulan
dbs4.ulan
dbs5.ulan

Destination DBS

dbs1.ulan
dbs3.ulan
dbs4.ulan
dbs5.ulan

☐ **disperse evenly on other dbs**

Source

Host from	DB indexes	Spot ids	User settings	Remove
dbs1.ulan	× 5000 × 5002	Comma-separated	Max users(int or %) any	🗑

Destination

Host To	DB indexes	Spot ids	Remove
dbs2.ulan	× 5006 × 5008	Comma-separated	🗑

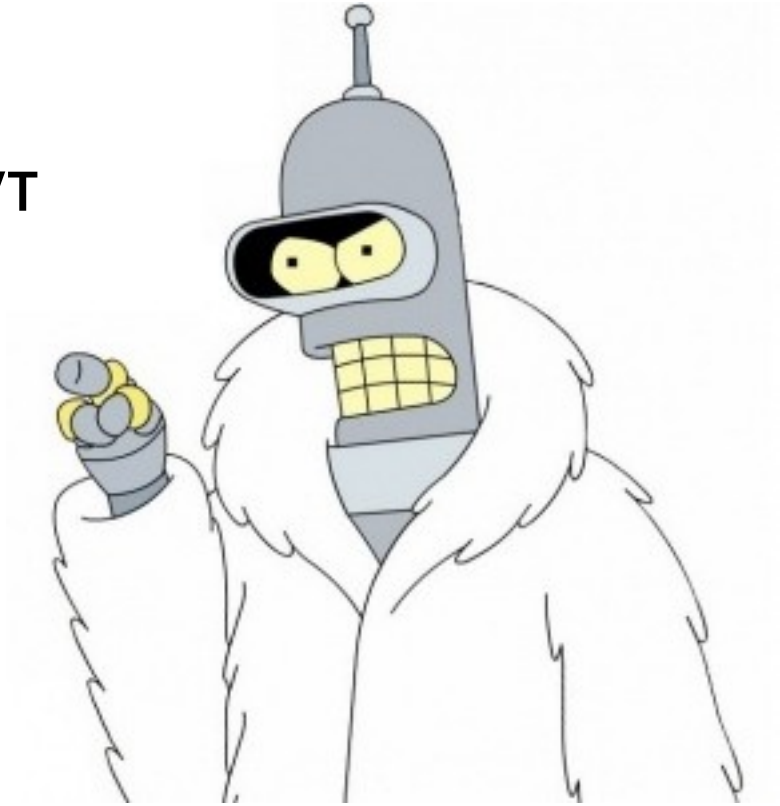
Submit

Есть над чем поработать!

- Блокировка пользователя
  - 75 перцентиль - 5 минут
  - 95 перцентиль - 10 минут
- Отмена миграции
- Сетевое разделение

Есть над чем поработать!

- Блокировка пользователя
  - 75 перцентиль – 5 минут
  - 95 перцентиль – 10 минут
- Отмена миграции
- ~~Сетевое разделение~~
  - У нас три аплинка!



Спасибо за внимание!
Ваши вопросы

krash@corp.badoo.com
krash3@gmail.com
facebook.com/alex.krash