

IMDG/GridGain

Промышленное применение

Михаил Гетманов, Сбербанк Технологии
Дмитрий Дрожжин, Сбербанк Технологии



Кто эти люди?

Михаил Гетманов



Последние 10 лет занимался разработкой решений для Телекома и около банковского сектора с использованием нетипичных для enterprise технологий. До СберТеха работал в ArtaSoftware.

Дмитрий Дрожжин



Разрабатываю IT системы для банков более 7 лет. До СберТеха работал над системами обработки трейдов в инвестиционных банках UBS и Deutsche Bank.

Создаем в Сбербанке систему хранения клиентских данных

Цели проекта

- Консолидация клиентских данных со всех систем банка
- Идентификация клиентов
- Дедупликация клиентских профилей
- Тиражирование “золотой записи” клиента

Дедупликация?

Что происходит, когда вы оформляете кредит?

Ваши персональные данные попадают к нам из АС "Кредиты".

Кредит погасили, можно преумножить сбережения!

Открываем депозитный счет.

Персональные данные снова попадают к нам из АС "Депозиты".

Наша задача – узнать вас во втором случае и объединить данные в единый профиль.

Сколько клиентов у Сбербанка?

- Более 110м
- Из них 30м регулярно пользуются онлайн сервисами (80% всех транзакций)
- Их число будет увеличиваться



Как работать с таким объемом данных?

Резервы вертикального масштабирования реляционных БД практически исчерпаны

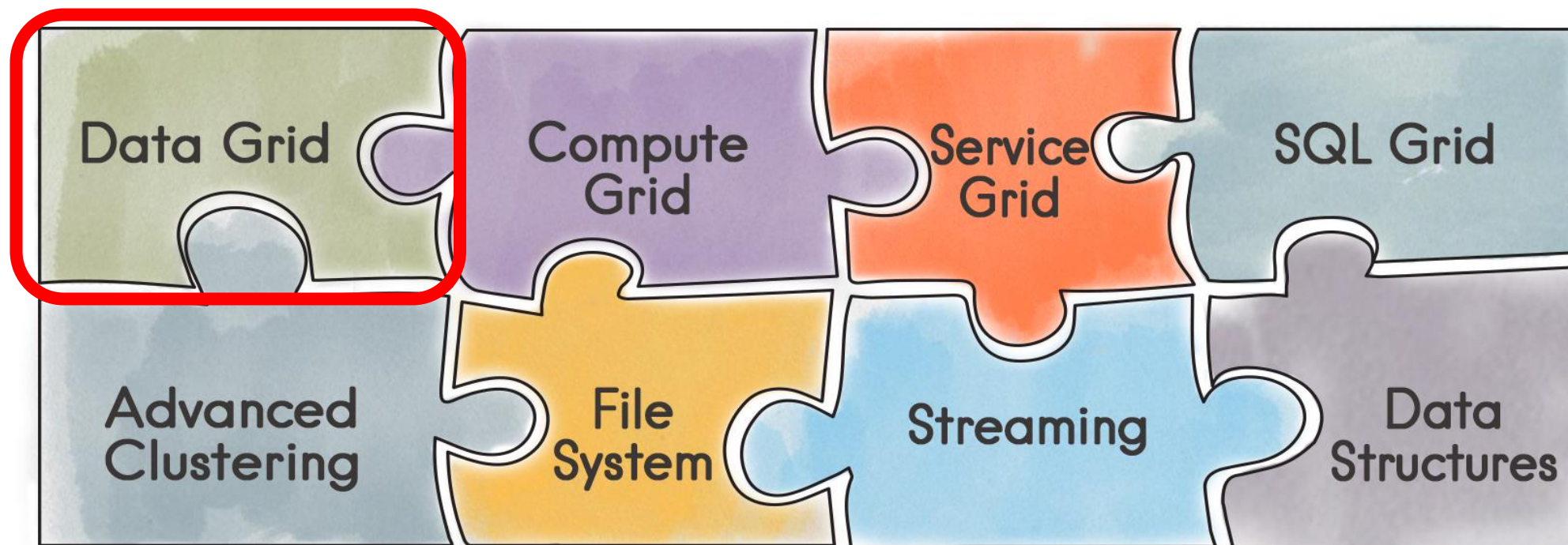
- В Сбербанке уже используется одно из самых производительных решений, которые можно купить
- И тем не менее его возможностей хватает только для текущего объема транзакций

Как это касается вас

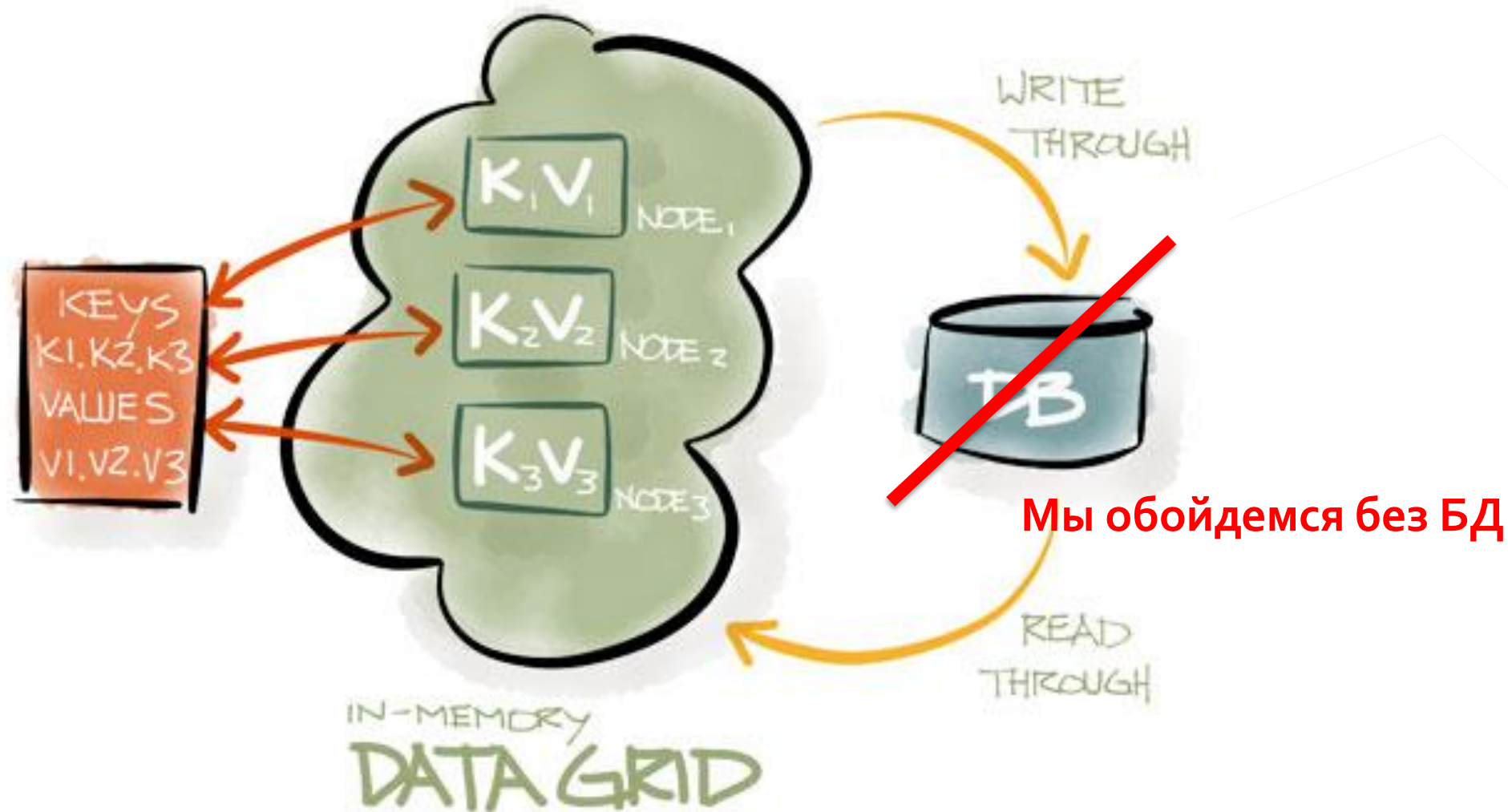
- Не проходят транзакции в магазине
- Долгая авторизация в сбербанк онлайн
- Длительная задержка при зачислении зарплаты
- В отделении вас обслуживают дольше

Горизонтальное масштабирование

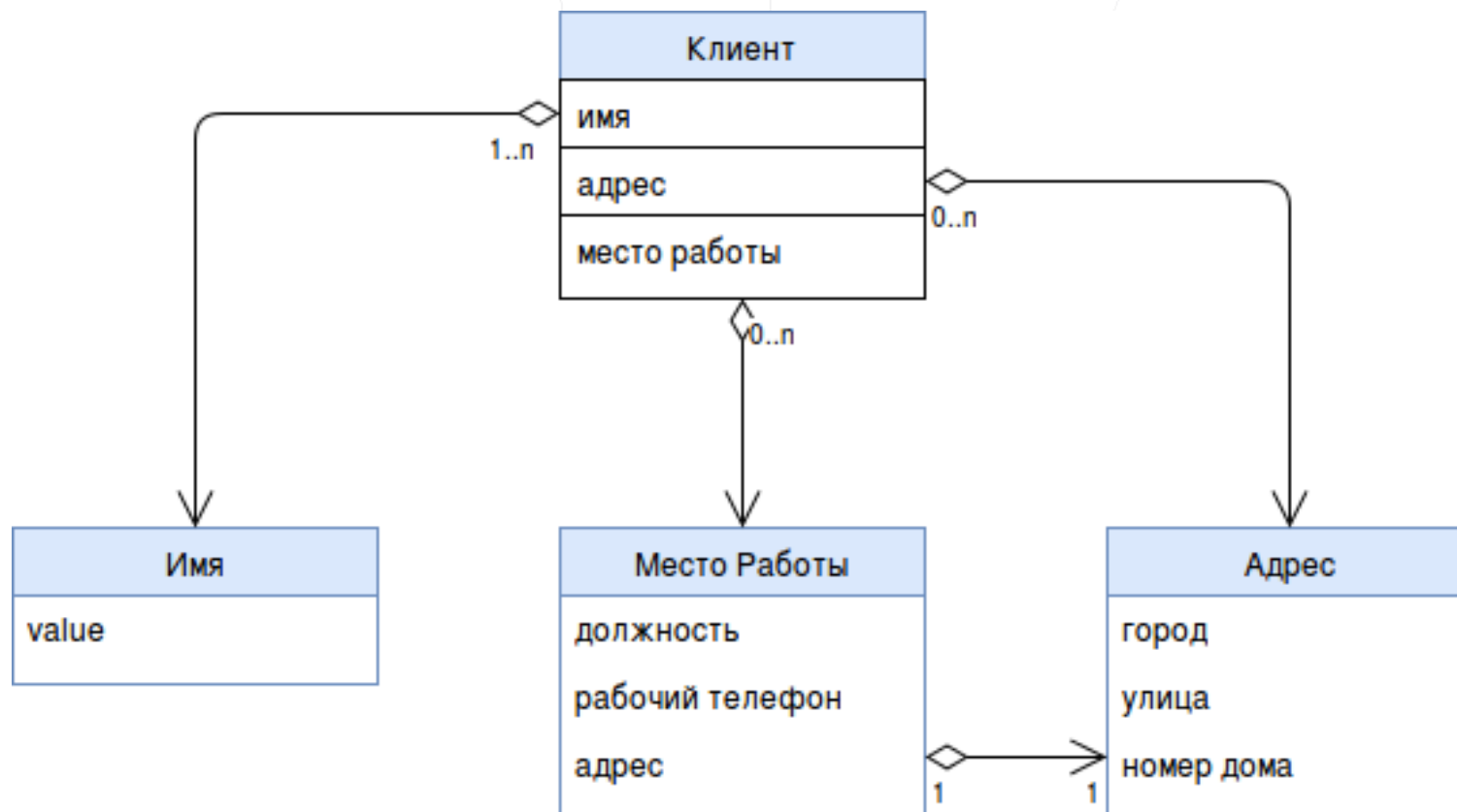
GridGain как основное хранилище данных



DataGrid для чайников



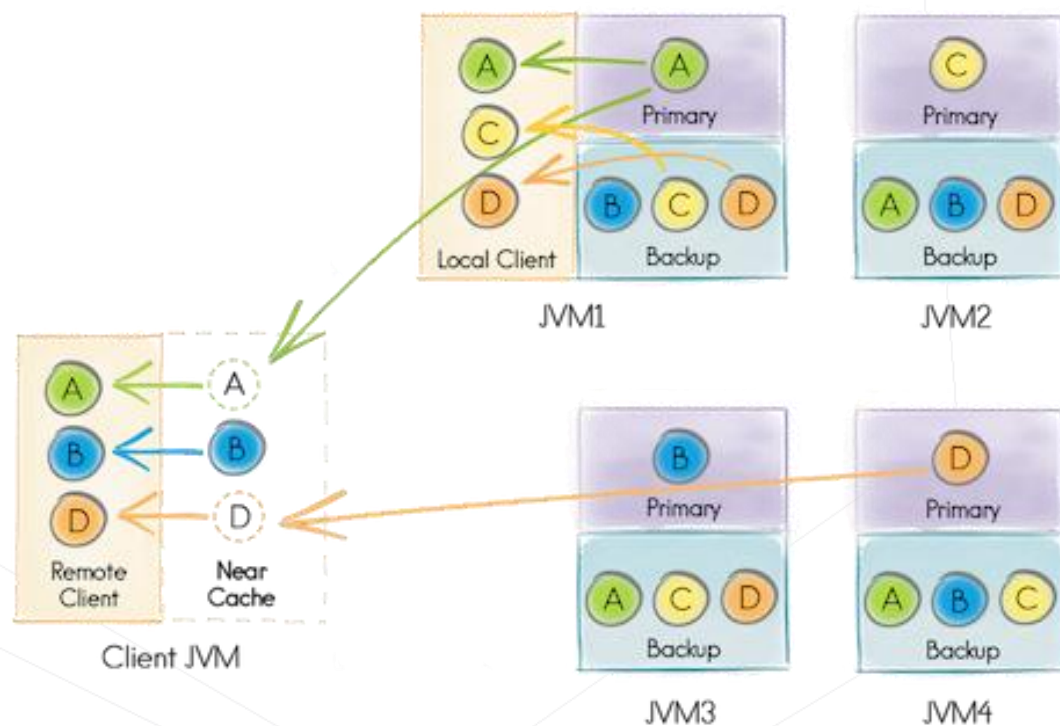
Как устроен клиент?



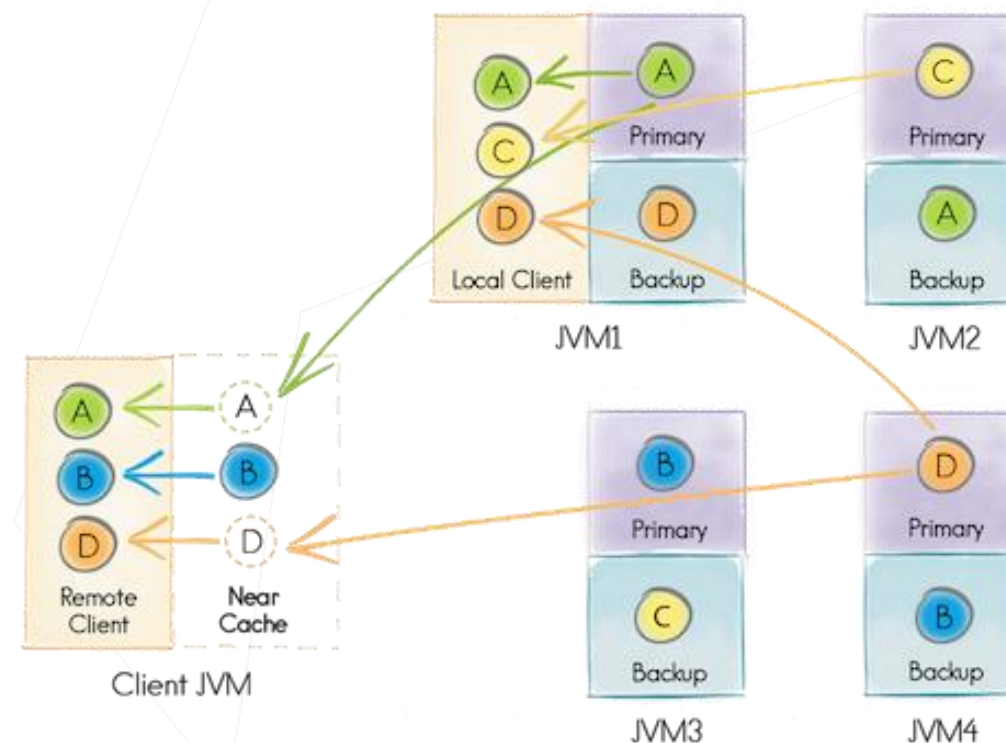
Disclaimer: на самом деле, все несколько сложнее

Где его можно сохранить в гриде?

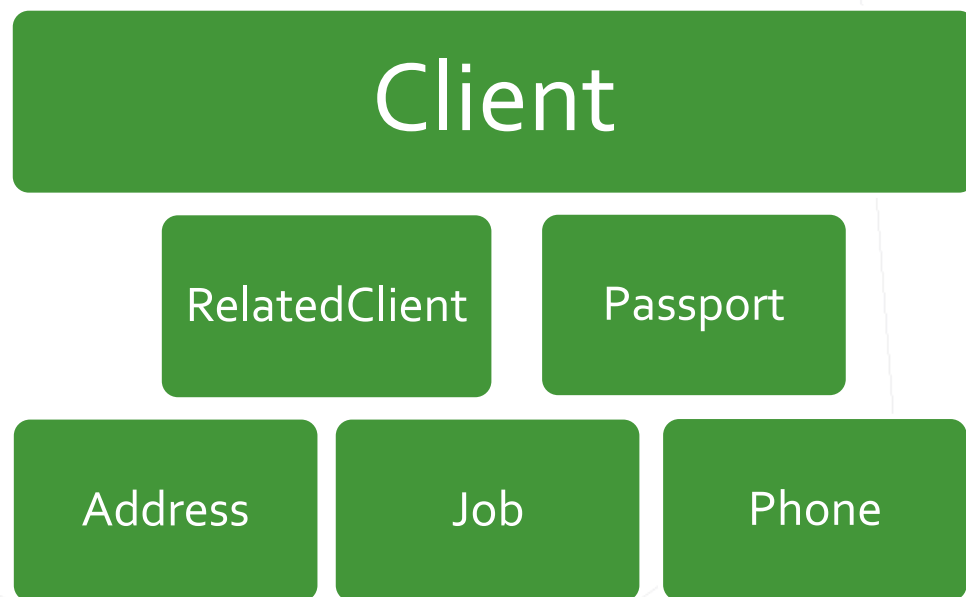
Replicated Cache



Partitioned Cache



Храним каждый элемент сущности клиента в отдельном кэше



Преимущества

- Частичное чтение и запись
- Поиск с помощью SQL запросов

Особенности

- Необходимость управления соразмещением данных
- Накладные расходы (2-10MB на кэш)

Весь клиент в одном кэше



CLIENT

Преимущества

- Компактное размещение
- Меньше накладных расходов

Особенности

- Нет частичного чтения и записи
- Нетривиальный поиск
- Не каждый клиент поместится на один узел

Что выбрать?

Взвесив все за и против, мы выбрали размещение клиента в нескольких кэшах. Для хранения служебной информации используем один кэш (например, история поисковых запросов).

Вам рекомендуем ориентироваться на ваши сценарии использования!

С гридом определились, что дальше?

Пишем наше распределенное приложение! Для этого нам нужны

- VIM / EMACS / IDEA / Eclipse
- Javac
- Возможность локально запустить приложение
- 20 секунд терпения. Столько стартует грид с инициализацией кэшей.
- 4GB “лишней” памяти! Потому что у нас 400 кэшей, на каждый из которых всегда резервируется 10 MB. **В версии 1.10.1.ea7 накладные расходы уменьшились до 2MB!**

Вымышленная история

Успех!

Наше приложение в продакшене.

Банк улучшает свои финансовые показатели и поглощает своего менее удачливого конкурента...

ИСТОРИЯ ОДНОГО ПОГЛОЩЕНИЯ

Начальные условия

- Нужно произвести слияние клиентских баз
- Существующая “сервис зона” должна принять новую партию клиентов

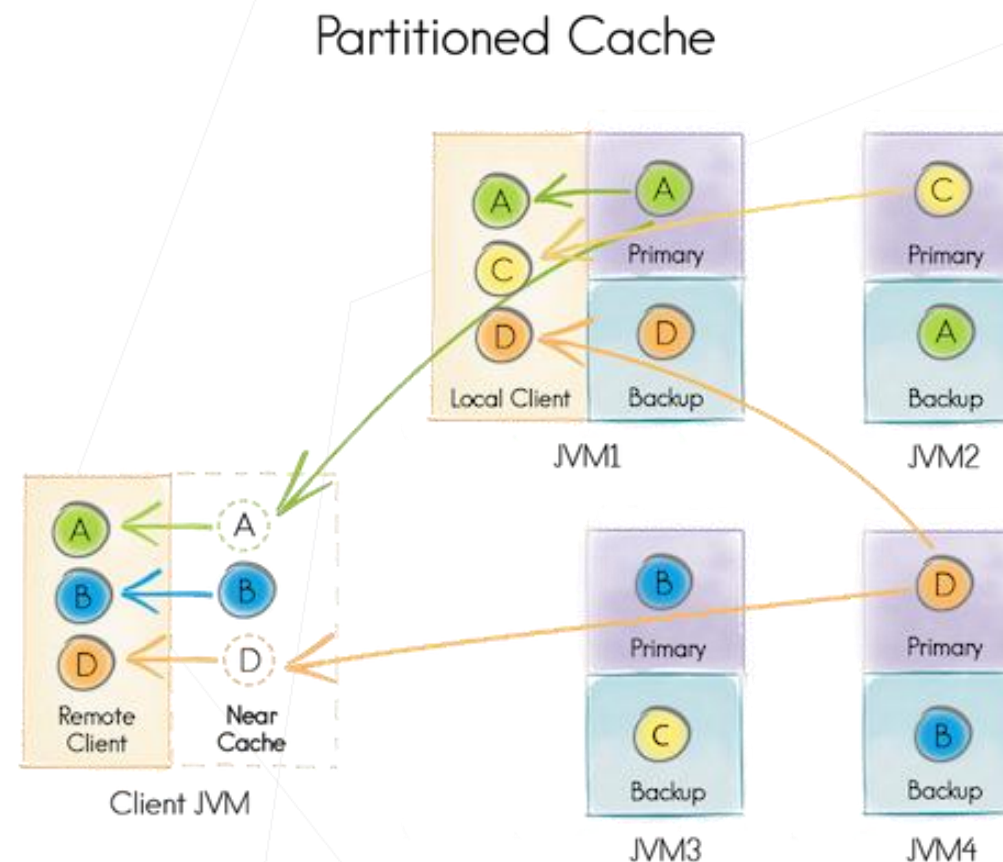
Зона обслуживания – группа клиентов, объединённых общим признаком. Для клиентов одной зоны действуют общие правила обслуживания – набор продуктов, тарификация и т. д.



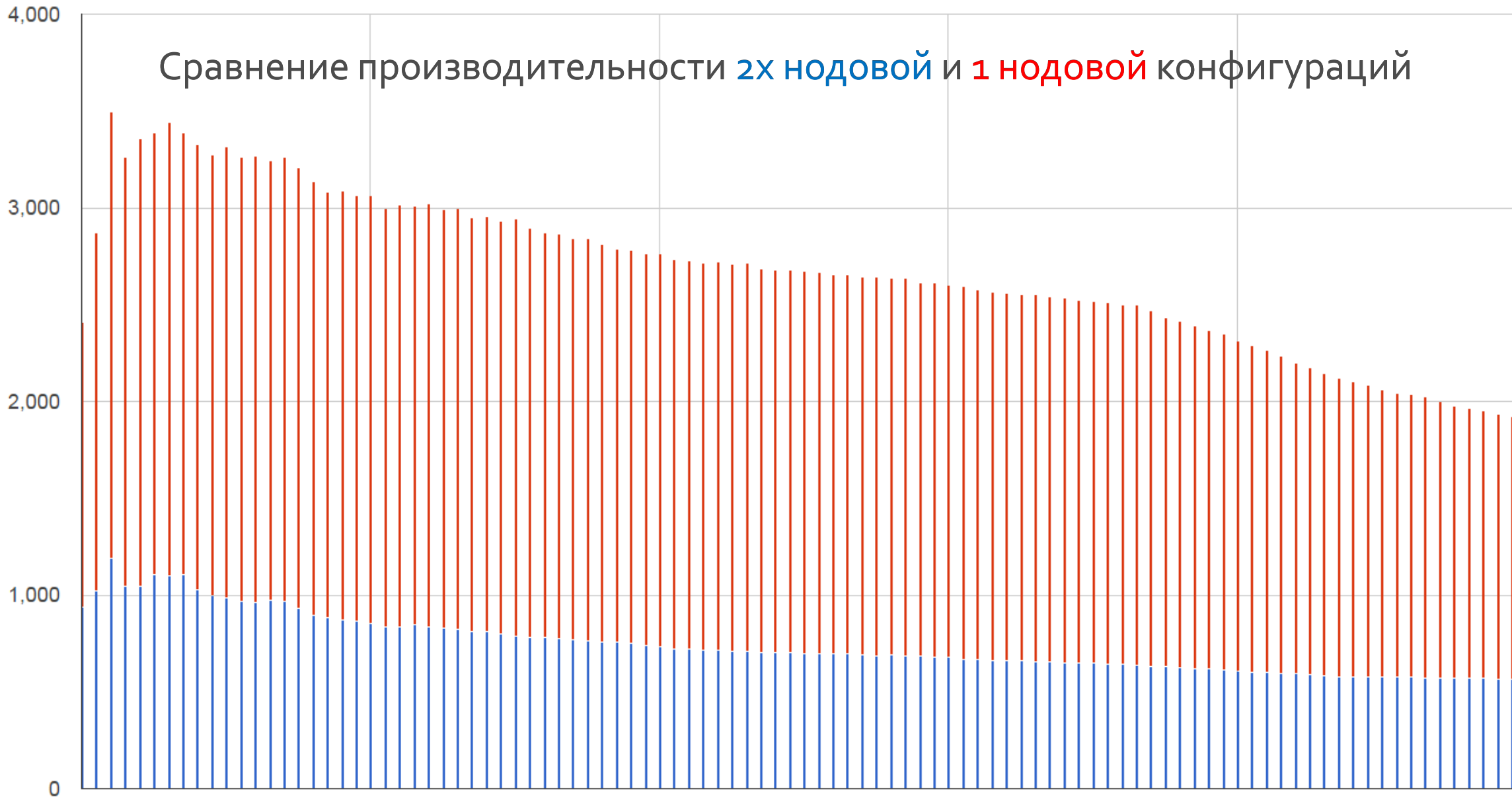
Решение: вводим новые сервера в грид!

Сложности масштабирования

- Необходимо предусмотреть возможность расширения грида на этапе создания affinity функции
- Ресурсоемкость ребалансировки



Сравнение производительности 2х нодовой и 1 нодовой конфигураций



Что делать и как жить

- Определитесь с целью масштабирования
- Масштабирование грида не всегда имеет следствием увеличение производительности приложения
- Перемещайте вычисления к данным
- Проводите оптимизации кода приложения на одной ноде
- Очень хорошо подумайте, если решили масштабироваться

Ребалансировка (физический смысл)

- Перераспределение партиций в гриде
- Изменение карты бэкапов
- Равномерное хранения данных на узлах

РЕБАЛАНСИРОВКА: ПРАВДА ЖИЗНИ

Потеря производительности



Антибонусы

- Потеря производительности на время ребалансировки (~50%)
- Утилизации сети
- Повышение утилизация диска и CPU
- * Транзакции с ошибками

Антибонусы, как пережить

- Потеря производительности на время ребалансировки (~50%)
- Утилизации сети
- Повышение утилизация диска и CPU
- * Транзакции с ошибками
- Контролировать причины
- Мониторить систему
- Не запускать параллельных ресурсоемких процессов
- Планировать изменение топологии

Пережили ребалансировку. На узлах равное количество данных.

ГРУЗИМ ДАЛЬШЕ!

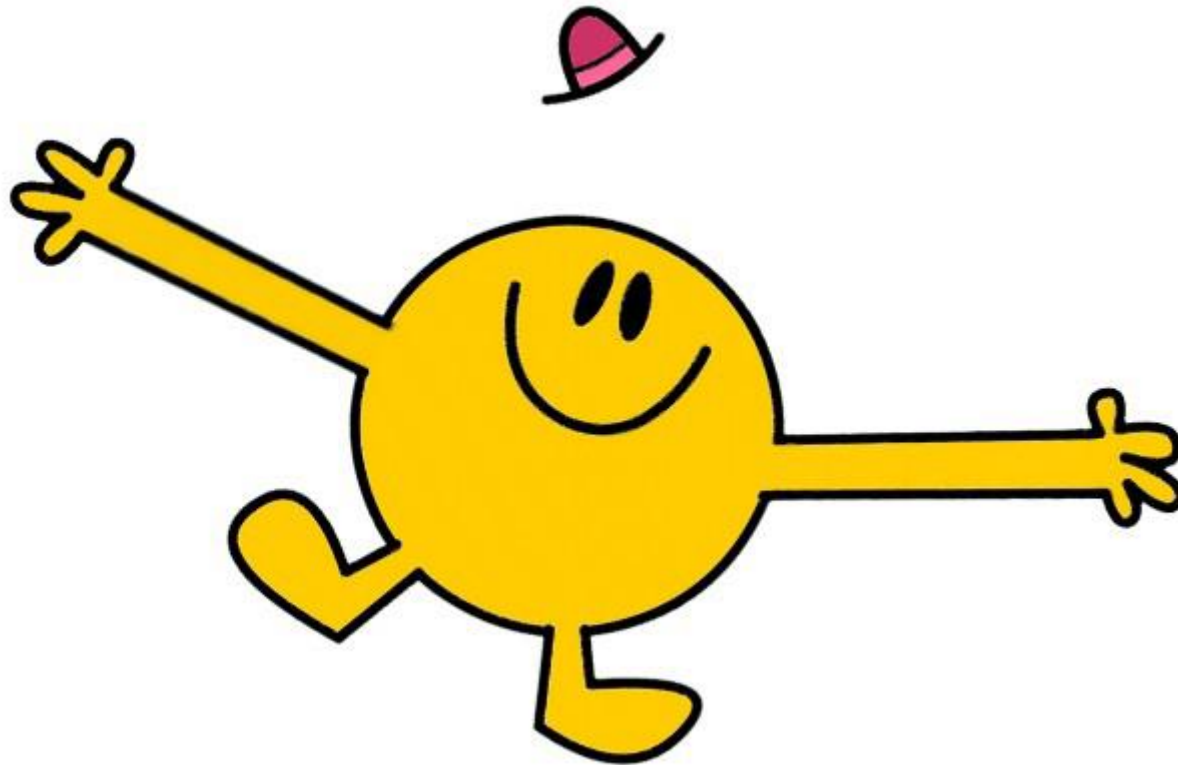
Дозагрузка клиентов

Физический смысл

- Проверяем существование клиента в нашем гриде
- Создаем или обновляем клиентский профиль

Позитив!

- Вероятность, что вас знают - выше
- Вас узнаем с новой стороны (актуальные телефоны, адреса и другие контактные данные)



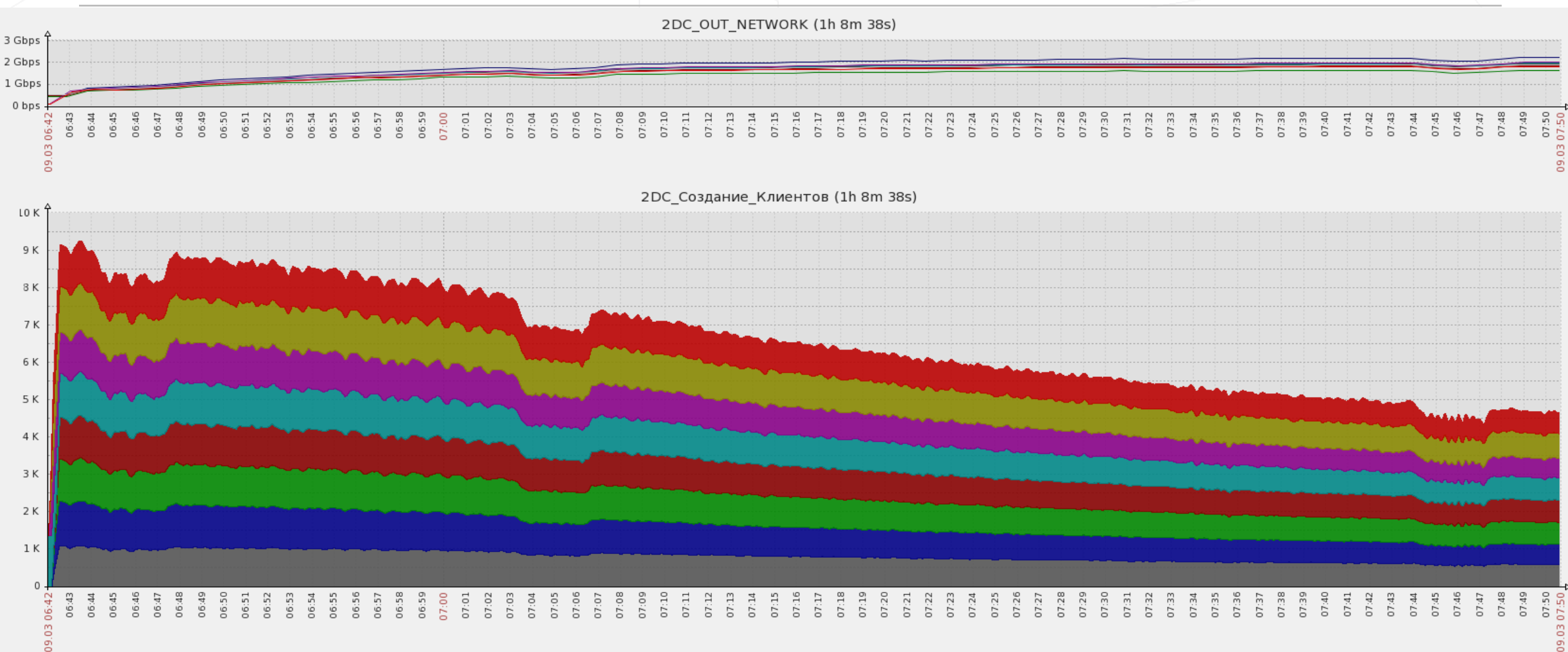
Как искать в гриде?

- Можно использовать SQL
- Сканирование (map reduce)
- Внешний индексатор
- Кастомный поиск

Кастомные глобальные индексы

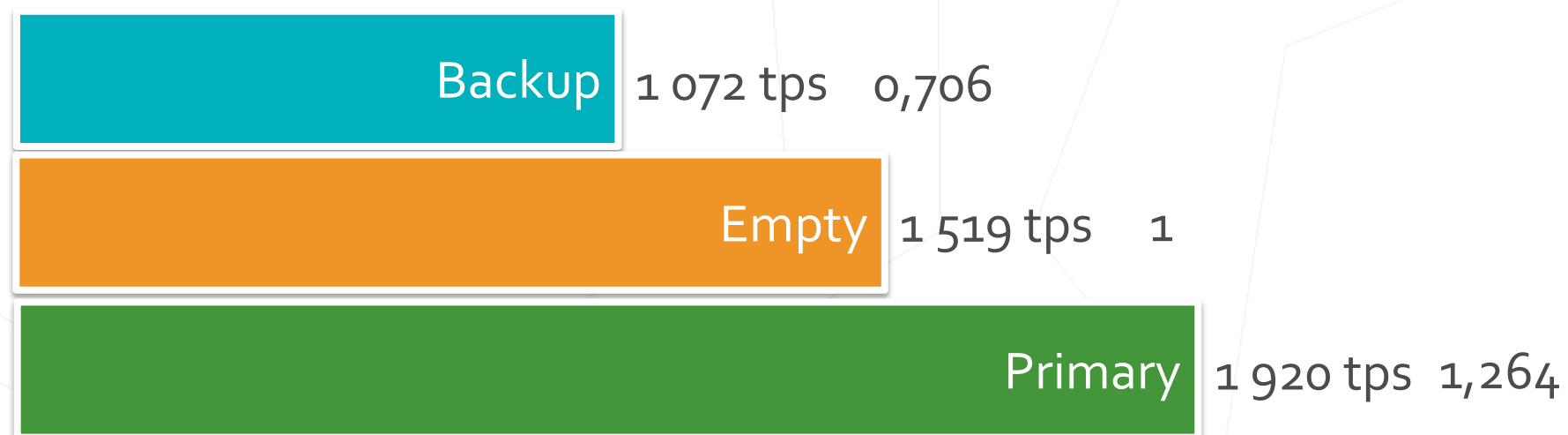
- Создается отдельный кэш на индексируемые поля
- Поиск объекта по индексу подразумевает несколько запросов: к кэшу-индексу, к основному кэшу

Неселективные значения в индексе



Маршрутизация запросов

А есть ли смысл?



ВЫВОД

- Избегай ребалансировку
- Контролируй ресурсы ноды
- Ожидай неуспешной транзакции
- Избегай неселективности индексов
- Маршрутизируй запросы