ПРОЕКТИРОВАНИЕ И РЕФАКТОРИНГ*, ВЫСОКАЯ ПРОИЗВОДИТЕЛЬНОСТЬ*, АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМ*, GOOGLE CLOUD PLATFORM*, БЛОГ КОМПАНИИ ИНФОПУЛЬС УКРАИНА

Вы — не Google

ПЕРЕВОД

tangro 12 июня 2017 в 12:26

Оригинал: Ozan Onay

Мы, программисты, иногда почему-то сходим с ума. Причём по каким-то совершенно нелепым причинам. Нам нравится думать о себе, как о супер-рациональных людях, но когда дело доходит до выбора ключевой технологии нового продукта, мы погружаемся в какое-то безумие. Вдруг оказывается, что кто-то слышал что-то об одной классной вещи, а его коллега читал комментарий о другой на Хабре, а третий человек видел пост в блоге о ещё чём-то похожем… и вот мы уже пребываем в полнейшем ступоре, беспомощно барахтаясь в попытках выбора между совершенно противоположными по своей сути системами, уже и забыв, что мы вообще пытаемся выбрать и почему.

Рациональные люди не принимают решения таким образом. Но именно так программисты часто решают использовать что-то вроде MapReduce.

Вот как комментировал этот выбор Joe Hellerstein своим студентам (на 54-той минуте):

> Дело в том, что в мире сейчас есть где-то 5 компаний, обрабатывающие данные подобных объёмов. Все остальные гоняют все эти данные туда-сюда, добиваясь отказоустойчивости, которая им на самом деле не нужна. Люди страдают гигантоманией и гугломанией где-то с середины 2000-ых годов: «мы сделаем всё так, как делает Google, ведь мы же строим один из крупнейших (в будущем) сервисов по обработке данных в мире!»

Сколько этажей в вашем датацентре? Google сейчас строит четырёхэтажные, как вот этот в Оклахоме.

Получить больше отказоустойчивости, чем вам на самом деле нужно, может показаться хорошей идеей. Но это только в том случае, если бы это получилось автоматически и бесплатно. Но это не так, у всего есть своя цена: вы не только будете пересылать

значительно больше данных между узлами (и платить за это), но и ступите с твёрдой земли классических систем хранения данных (со всеми их транзакциями, индексами, оптимизаторами запросов и т.д.) на зыбкую почву слабой инфраструктуры новомодных систем. Это важный шаг (и часто — шаг назад). Сколько пользователей Hadoop сделали его сознательно? Сколько из них со временем могут сказать, что это был выверенный и мудрый шаг?

MapReduce и Hadoop в этом плане являются лёгкими мишенями для критики, ведь даже последователи карго-культа со временем признали, что их обряды как-то не очень работают для привлечения самолётов. Но это наблюдение можно и обобщить: если вы решаете использовать какую-то технологию, созданную большой корпорацией, вполне может быть, что вы не пришли к этому решению сознательно; вполне возможно, что к этому привела какая-то мистическая вера в то, что можно имитировать поведение гигантов и таким образом достичь тех же высот.

Да, это ещё одна «анти карго-культ» статья. Но подождите! У меня ниже есть полезный чек-лист, который поможет вам принимать более взвешенные решения.

Клёвая технология? Ну, давайте заценим.

В следующий раз, когда вы вдруг обнаружите совершенно обалденную технологию, на которой стоит вот прямо сегодня переписать всю вашу архитектуру, я попрошу вас сделать паузу:

1. Задумайтесь о том, какую проблему вам нужно решить в вашем продукте. **Поймите её до конца**. Вашей задачей должно быть решить эту проблему с помощью каких-то инструментов, а не

решить «какую-то» проблему, которую может решить понравившийся вам инструмент.

2. Выберите **несколько** потенциальных инструментов. Не выбирайте себе одного «любимчика», пока не будете иметь списка из нескольких альтернатив!

3. Для каждого кандидата найдите его "**Главный Документ**" и прочтите его. Не комментарии или переводы, а «тот самый» документ.

4. Определите **исторический контекст**, который привёл к созданию данного инструмента таким, каким он был создан.

5. Идеалов не бывает. Взвесьте **преимущества** и недостатки каждого инструмента. Поймите, на какие компромиссы пришлось пойти его создателям, чтобы достичь главных заявленных ими целей.

6. **Думайте**! Трезво и честно ответьте себе на вопрос, совпадают ли ваши цели и приоритеты с целями и приоритетами создателей тех инструментов, которые вы планируете использовать. Какой новый факт об этом инструменте мог бы заставить вас отказаться от его использования? Например, насколько должен отличаться масштаб ваших задач от тех, на которые рассчитан инструмент, чтобы заставить вас задуматься о нерациональности его применения?

А ещё вы — не Амазон

Вышеуказанную методику достаточно легко применять на практике. Совсем недавно я обсуждал с одной компанией их планы использовать Cassandra для нового (достаточно нагруженного операциями чтения) проекта. Я прочёл оригинальный документ, в котором описывались принципы работы Dynamo и, зная, что Cassandra достаточно близка к Dynamo, понял, что эти системы строились на принципе приоритизации операций записи (Амазон хотел, чтобы операция «добавить в корзину» абсолютно всегда проходила успешно). И они даже добились этого, пожертвовав, однако, гарантированной консистентностью и функциональностью традиционных реляционных баз данных. Но компания, с которой я обсуждал применения этого решения, не ставила запись данных в приоритеты (все данные приходили одним блоком раз в сутки), а вот консистентность и производительное чтение были как-раз очень нужны.

Амазон продаёт много разных товаров. Если операция «добавить в корзину» вдруг начнёт работать нестабильно — они потеряют много денег. У вас та же ситуация?

Компания, которая хотела внедрить у себя Cassandra, делала это потому, что один из запросов в их PostgreSQL выполнялся несколько минут и они считали, что упёрлись в аппаратные возможности платформы. После буквально пары уточняющих вопросов, оказалось, что перенос их базы на SSD ускорял этот запрос на 2 порядка (до нескольких секунд). Это всё ещё было медленно, конечно, но просто оцените, как переход на SSD (секундное дело), смог ускорить узкое место где-то в 100 раз. Без всяких фундаментальных переделок архитектуры. Стало совершенно ясно, что переход на Cassandra — не верный путь.

Здесь требовался некоторый тюнинг существующего решения, возможно — перемоделирования схемы базы данных, но определённо не попытки решить проблему ЧТЕНИЯ инструментом, созданным для решения проблем ЗАПИСИ.

К тому же, вы — не LinkedIn

Я был очень удивлён, когда узнал, как одна моя знакомая компания решила построить свою архитектуру вокруг Kafka. Это показалось мне странным, поскольку их основной задачей был процессинг нескольких десятков (в хороший день — нескольких сотен) достаточно важных транзакций. Для такой входной нагрузки подошло бы даже решение "*бабушка, вручную записывающая операции в бумажный блокнот*". Kafka, чтобы вы понимали, был создан для сбора всей аналитической информации LinkedIn (все вот эти сообщения, запросы, группы, чаты, новости, оценки и т.д.). Огромное количество данных. Не могу оценить их объёмы сейчас, но несколько лет назад LinkedIn называл цифру порядка 1 триллиона событий в день, с отдельными пиками до 10 миллионов в секунду. Да, я понимаю, что и 10 транзакций в день тоже с его помощью можно успешно обработать. Но зачем это делать? Разница теоретической нагрузки и практической необходимости составляет 10 ПОРЯДКОВ!

Наше Солнце, например, всего на 6 порядков больше Земли.

Да, вполне возможно, что инженеры этой компании рационально оценили все преимущества Kafka и выбрали этот инструмент по каким-то неизвестным мне причинам, не связанным с нагрузками. Но более вероятно, что они просто подпали под влияние какого-то восторженного отзыва, статьи, мнения сообщества или что-то ещё, кричавшее им «бери Kafka»… Нет, ну вы подумайте, 10 порядков!

Я уже говорил, что вы — не Амазон?

Ещё более популярной вещью, чем распределённые базы данных Амазона, является их архитектурный подход для масштабирования: сервис-ориентированная архитектура (SOA). В 2001 году Амазон осознал, что не может эффективно

масштабировать нагрузки на свой front end, и, работая над решением этой проблемы, пришел к SOA. Эта история успеха передавалась устно и письменно от одного инженера к другому и вот уже мы приходим к ситуации, когда каждый первый стартап из трёх программистов и нуля пользователей начинает свой жизненный путь с того, что разбивает свою домашнюю страницу на наносервисы. К тому времени, когда Амазон осознал необходимость перехода на SOA, у них было 7800 сотрудников и они продавали товаров на 3 миллиарда долларов в год.

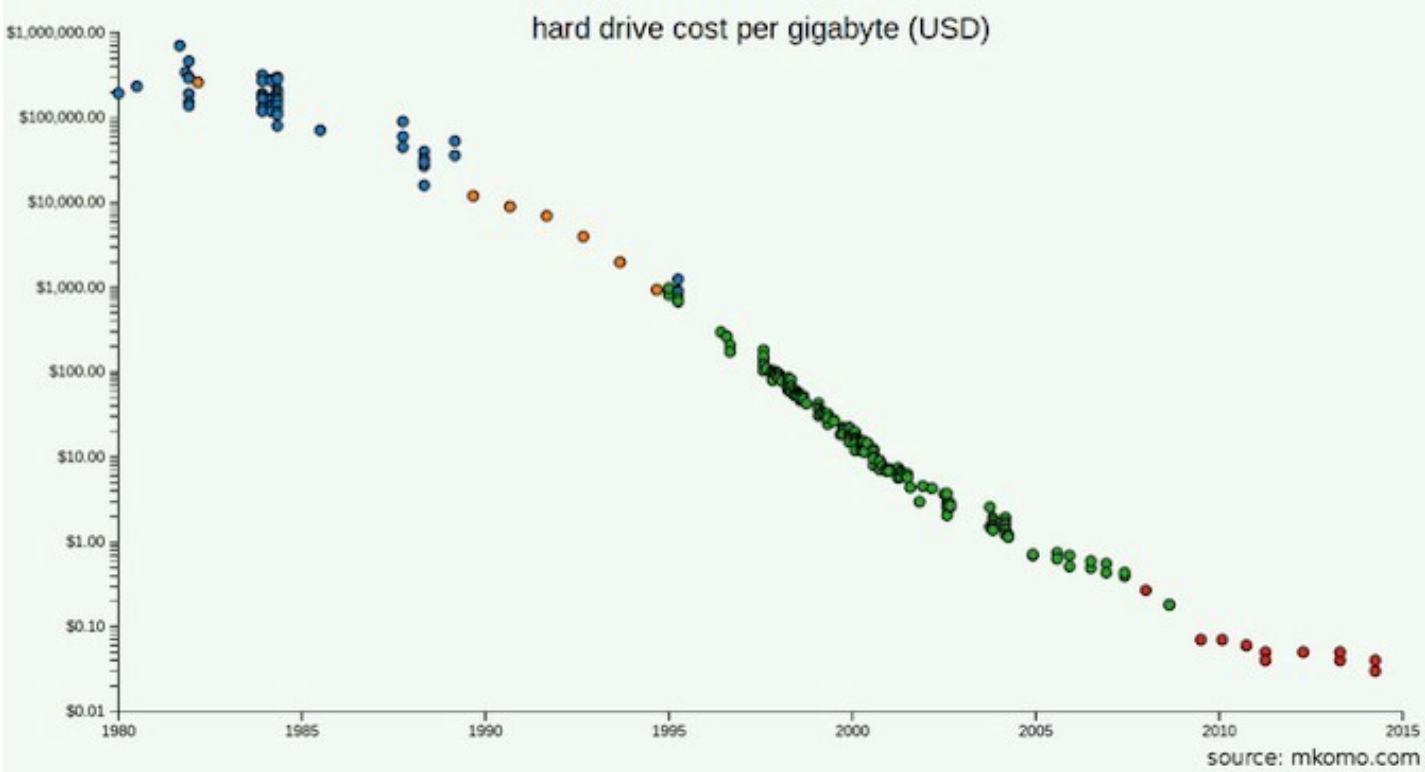
Вот этот зал вмещает 7000 людей. А у Амазона было 7800, когда понадобился переход на SOA.

Я не говорю, что вам нужно ждать момента найма 7800-го сотрудника для внедрения SOA. Просто задайте себе вопрос — это вот именно та проблема, которую нужно решать сейчас? Других уже нет? Если вы мне скажете, что это действительно так и ваша компания из 50 человек вот прямо сейчас упирается именно

в отсутствие SOA, то как вы объясните тот факт, что есть в десятки раз большие компании, которые от этого нисколько не страдают?

И даже Google — не Google

Использование хорошо масштабирующихся инструментов, вроде Hadoop или Spark может быть очень интересным. Однако, на практике классические инструменты лучше подходят для обычных и даже больших нагрузок. Иногда кажущиеся «большими» объёмы могут даже полностью поместиться в ОЗУ одного компьютера. Вы знали, что уже сегодня можете получить терабайт ОЗУ за примерно 10000 долларов? Даже если у вас есть целый миллиард пользователей (а у вас его нет), то за эту цену вы получите целый килобайт данных в ОЗУ на каждого из них. Очень быстро доступный килобайт. Возможно, этого недостаточно для ваших задач и придётся что-то читать\писать с диска. Но сколько это будет дисков — вот прямо тысячи? Вряд ли. Вам не понадобятся решения вроде GFS и MapReduce, которые создавались, на минуточку, для хранения поискового индекса ВСЕГО ИНТЕРНЕТА.



Место на жестких дисках сегодня стоит значительно дешевле, чем в 2003 году, когда было опубликовано описание GFS.

Возможно, вы читали документацию по GFS и MapReduce. Тогда вы можете помнить, что основной проблемой, которую пытался решить Google, была не вместимость, а пропускная способность. Они построили распределённую систему, чтобы получить доступ к нужным данным быстрее. Но сегодня на дворе 2017 год и пропускная способность устройств выросла. Учтите, что вам не понадобится обрабатывать столько же данных, сколько обрабатывает Google. Так, может быть, достаточно будет купить жестких дисков получше (возможно, SSD) и этого хватит?

Возможно, вы рассчитываете со временем вырасти. Но считали ли вы, на сколько именно? Будете ли вы аккумулировать данные быстрее, чем будет падать стоимость SSD? Насколько вашему бизнесу нужно будет вырасти до того момента, когда все ваши

данные перестанут помещаться на одной физической машине? В 2016 году система Stack Exchange, обслуживающая 200 миллионов запросов в сутки, использовала всего 4 SQL сервера: один для Stack Overflow, один для всего остального и две реплики.

Вы можете пройти по указанному мной выше чек-листу и всё равно остановиться на Hadoop или Spark. И это даже может быть верным решением. Важно, однако, понимать, что верное здесь и сейчас решение не обязательно останется таким надолго. Google знает это хорошо: как только они решили, что MapReduce недостаточно хорошо работает для построения поискового индекса, они перестали его использовать.

Прежде всего поймите вашу проблему

Эта мысль не моя и она далеко не нова. Но, возможно, в интерпретации этой статьи, вместе с чек-листом выше она достучится до вашего сердца. Если нет, вы можете просмотреть такие материалы, как Hammock Driven Development, How to Solve It или The Art of Doing Science and Engineering — во всех них проскакивает один и тот же призыв думать, пытаться понять проблему перед тем, как героически бросаться её решать. В книге G. Polya есть такая фраза:

> Глупо пытаться найти ответ на вопрос, которого ты не знаешь.
> Грустно работать над тем, что не решает твою проблему.

Похожие публикации

Эмулятор Bluestacks + Eclipse: ускоряем отладку и тестирование Android-приложений

38

tangro • 26 июля 2012 в 13:27

Как решить проблему конфликта приложений

20

ИЗ ПЕСОЧНИЦЫ

allexb • 14 ноября 2011 в 13:48

Особенности подготовки инсталляции приложения для автоматической (unattended) установки в Windows OS

19

Infopulse_Ukraine • 8 ноября 2011 в 17:42

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на

14

русском языке

alexkuku • вчера в 16:12

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада 4

lahmatiy • вчера в 13:05

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе 25

ПЕРЕВОД

Smileek • вчера в 10:32

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации 2

ПЕРЕВОД

ru_vds • вчера в 12:04

Как адаптировать игру на Unity под iPhone X к апрелю 0

P1CACHU • вчера в 16:13

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни 33

HostingManager • вчера в 13:49

Обзор рынка моноколес 2018

70

lozga • вчера в 06:58

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

40

alizar • вчера в 15:14

Стивен Хокинг и его работа — что дал ученый человечеству?

8

marks • вчера в 14:46

Sunlike — светодиодный свет нового поколения

17

AlexeyNadezhin • вчера в 20:32