Распределенные системы

План

- WTF
- Распределенные вычисления
- Немножко о ZeroMQ

WTF


РАСПРЕДЕЛЕННЫЕ
СИСТЕМЫ
ПРИНЦИПЫ И ПАРАДИГМЫ
Э. ТАНЕНБАУМ, М. ван СТЕЕН
ozon.ru
ПИТЕР

WTF

Распределенная система — это набор независимых компьютеров, представляющийся их поль-зователям единой объединенной системой

Эмпирические свойства

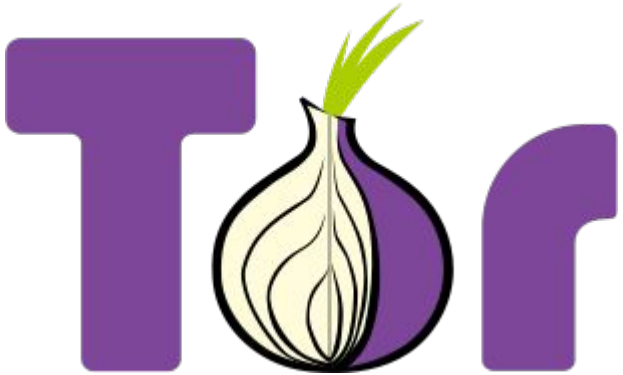
- Устойчивость при отказе одного из элементов
- Неоднородность элементов, конфигурация системы может изменяться в процессе функционирования
- Ни один из элементов не обладает полными знаниями о всей системе в целом

Что в этом хорошего?

- Выход из строя одного элемента системы не приводит к ее отказу в целом
- Масштабируемость
- Доступность

WTF

- DNS
- CDN
- P2P
- TOR
- AWS
- ...



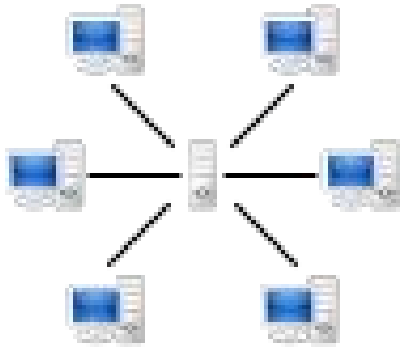
Кто и когда

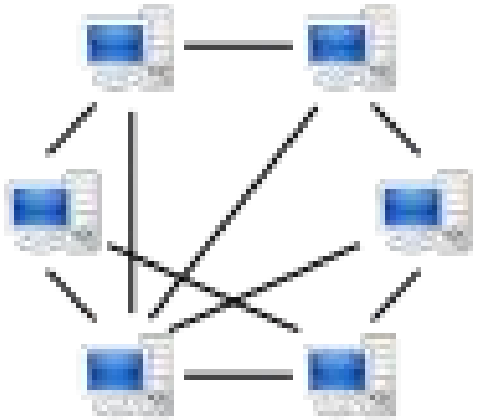
- Первые исследования в области начались в 1960х годах прошлого века
- 1960-1970 ARPANET, EMAIL
- 1970-1980 USENET, FIDONET
- И покатилось

Основные архитектуры

- Клиент - сервер
- Кластерная архитектура
- Peer-2-Peer

Versus battle





Versus battle

+ Большая централизованность
+ Более легкий менеджмент данных
+ Возможность делать бэкапы
+ Легче

- Меньшая устойчивость
- Необходимость высококвалифицированного персонала
- Ограниченность в производительности

ACID

- Атомарность - все транзакции атомарны
- Согласованность - каждая успешная транзакция вносит только допустимые результаты
- Изолированность - результат параллельных транзакций не зависит друг от друга
- Долговечность - изменения, созданные успешной транзакцией, не исчезают после возможных сбоев

в условиях распределенных систем выполнения требований становится куда сложнее

BASE

- Базовая доступность - сбои в отдельных узлах приводят только к ограниченной потери работоспособности системы в целом
- Неустойчивое состояние - возможность жертвовать долговечностью не в критических местах
- Согласование в конечном счете - данные могут быть противоречивы в определенный промежуток времени. В обозримое время должно произойти их согласование

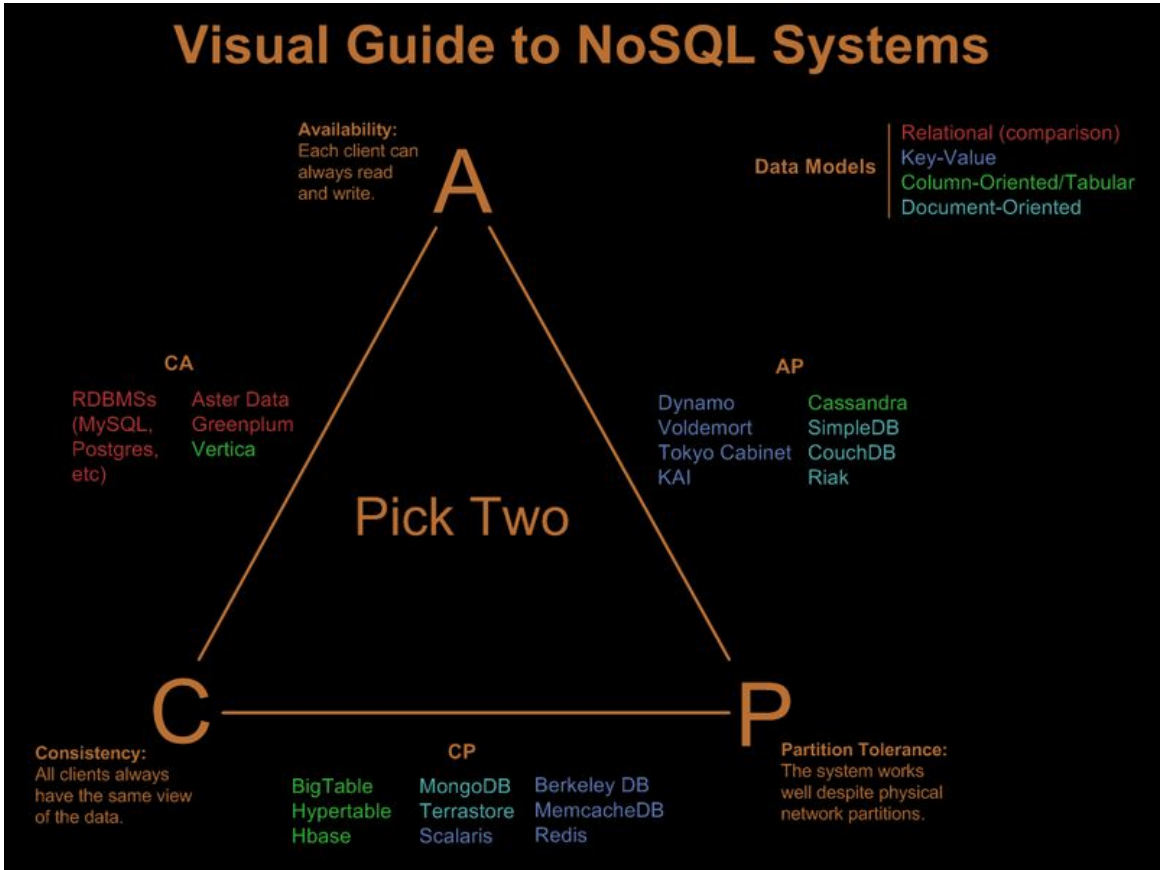
значительное усложнение процесса создания распределенных систем

CAP теорема (Брюера)

- Согласованность - во всех элементах (узлах) в один момент времени данные не противоречат друг другу
- Доступность - любой запрос к завершается корректным откликом
- Устойчивость - разделение на несколько изолированных секций не приводит к некорректности отклика от каждой из секций

возможны только 2 из 3

CAP теорема (Брюера)



Проблема

Зачем вообще нам нужны эти сложности, ведь делать все в рамках одного процесса проще?

Проблема

Физические ограничения железа

- Память
- Вычислительные мощности

Pipeline архитектура

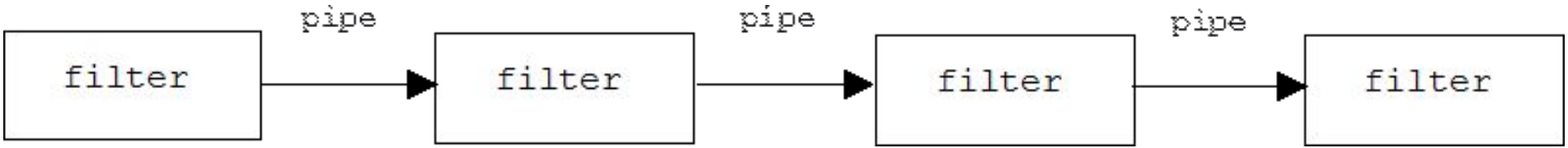
Задача:

- Есть некий поток данных
- Над ним необходимо совершать много различных затратных действий
- Время выполнения каждого действия значительно больше времени

Pipeline архитектура

cat usernames.txt | grep “your mom” | sort > moms

Pipeline архитектура



Межпроцессное взаимодействие

Как осуществлять передачу данных между процессами?

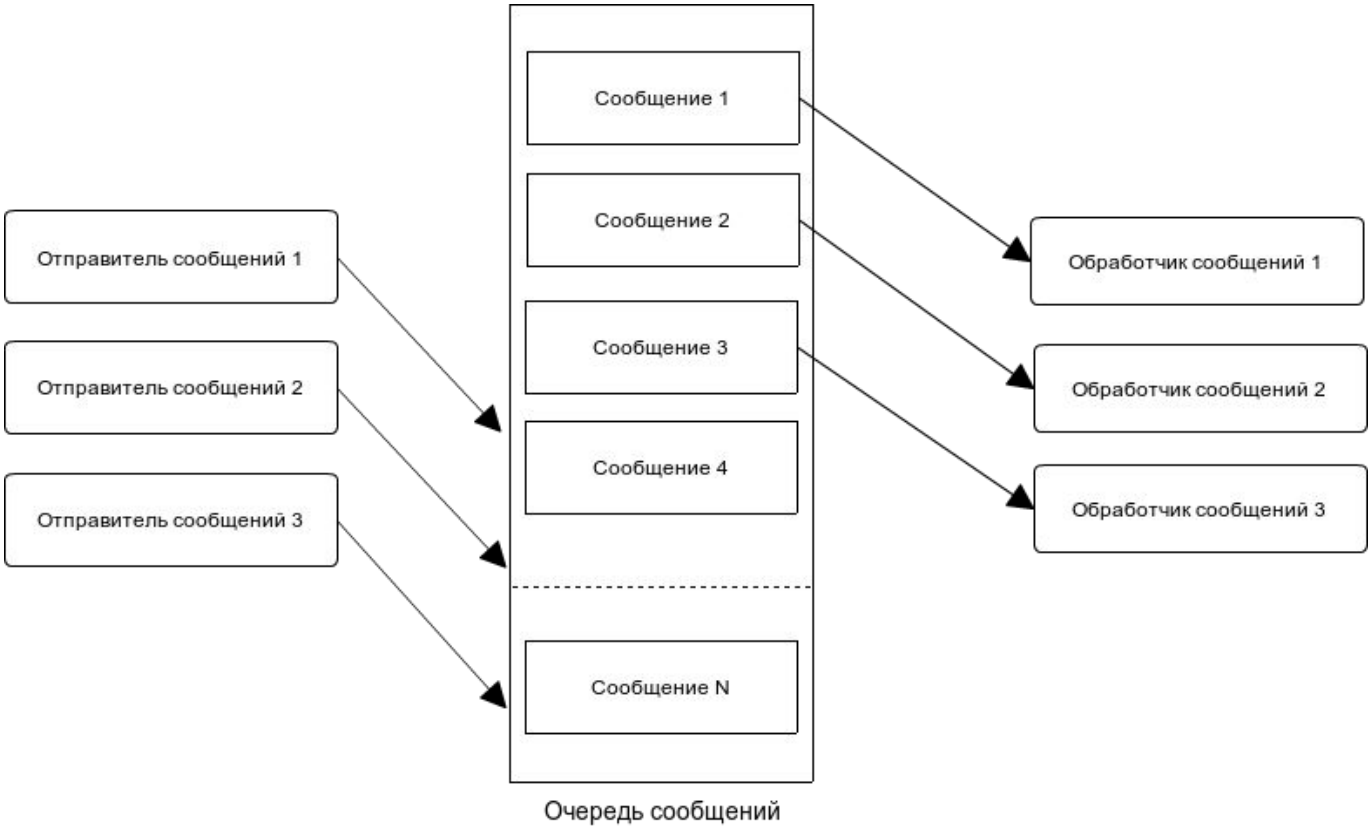
IPC

- Файлы
- Сигналы
- Pipes
- Семафоры
- Общая память
- Сокеты

IPC

- Файлы
- Сигналы
- Pipes
- Семафоры
- Общая память
- ***Сокеты***

Очереди сообщений



Очереди сообщений

- Масштабируемость
- Отказоустойчивость
- Буферизация
- Гарантированный порядок сообщений
- Гарантированная доставка сообщений
- Асинхронность

Очереди сообщений

Два относительно стандартизованных текстовых протокола, работающих на уровне HTTP

- Advanced Message Queuing Protocol
- Simple Text Oriented Messaging Protocol

- MQTT
- IETF CAP
- XMPP

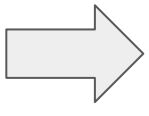
Очереди сообщений

- RabbitMQ
- ZeroMQ
- IronMQ
- OpenMQ
- Apache Kafka
- ...

MapReduce


Google
ИСТОРИЯ
Прорыв в духе времени
Дэвид А. Вайз
при участии Марка Малсида

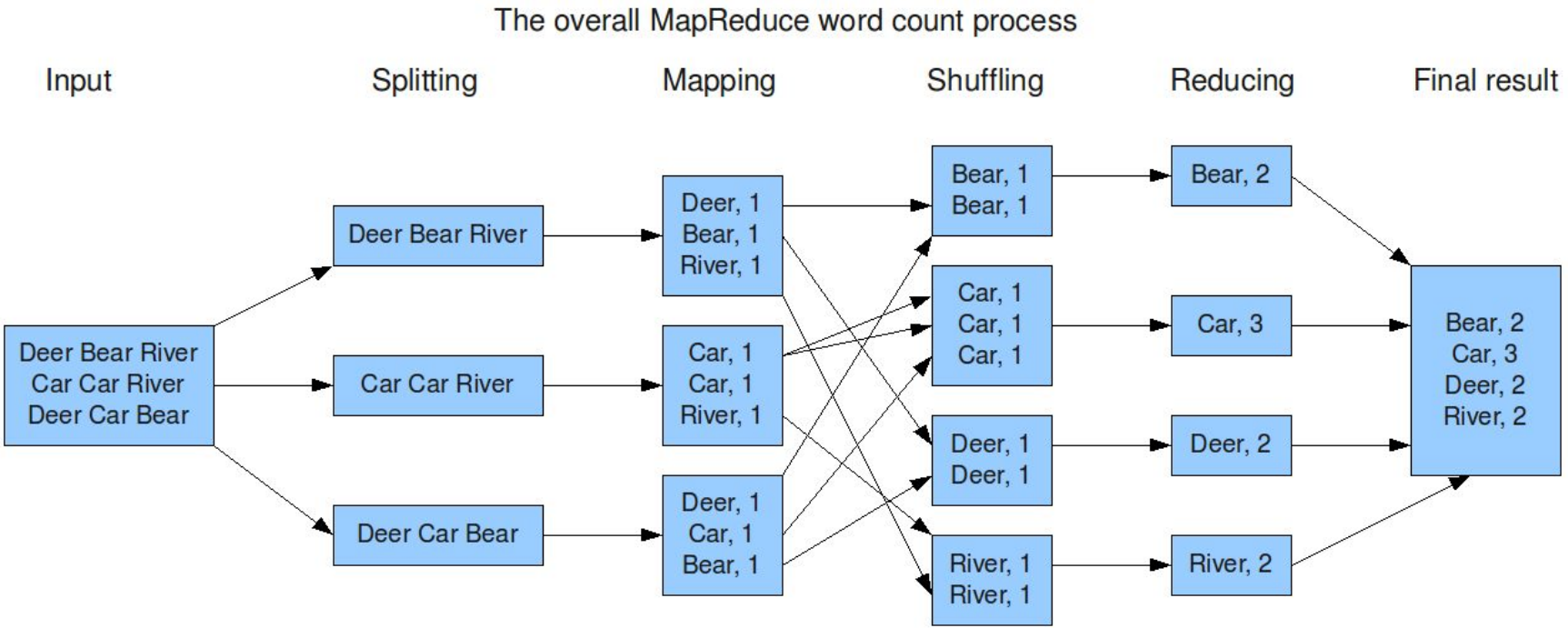
MapReduce



MapReduce

- MapReduce: Simplified Data Processing on Large Clusters, 2004
- По состоянию на 2014ый год Google отказался от концепции MapReduce
- Самая популярная open-source реализация - Apache Hadoop

MapReduce

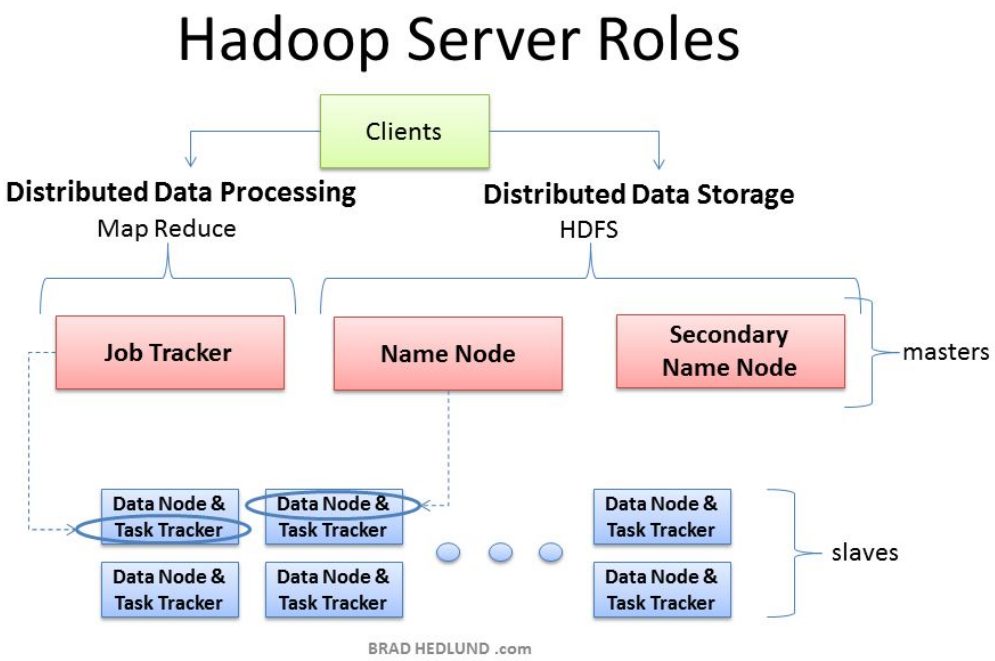

The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Apache Hadoop

- Написан на Java
- Предлагает интерфейсы для запуска произвольных задач
- Обладает хорошо развитой сопутствующей инфраструктурой (сетевая файловая система, хранилища данных, вспомогательные вещи для скриптинга)
- Обладает ОГРОМНЫМ количеством архитектурных недостатков
- Достаточно сложен в освоении, настройке и управлении

Apache Hadoop

Apache Hadoop



Apache Hadoop

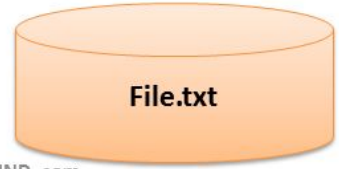
Typical Workflow

- Load data into the cluster (HDFS writes)
- Analyze the data (Map Reduce)
- Store results in the cluster (HDFS writes)
- Read the results from the cluster (HDFS reads)

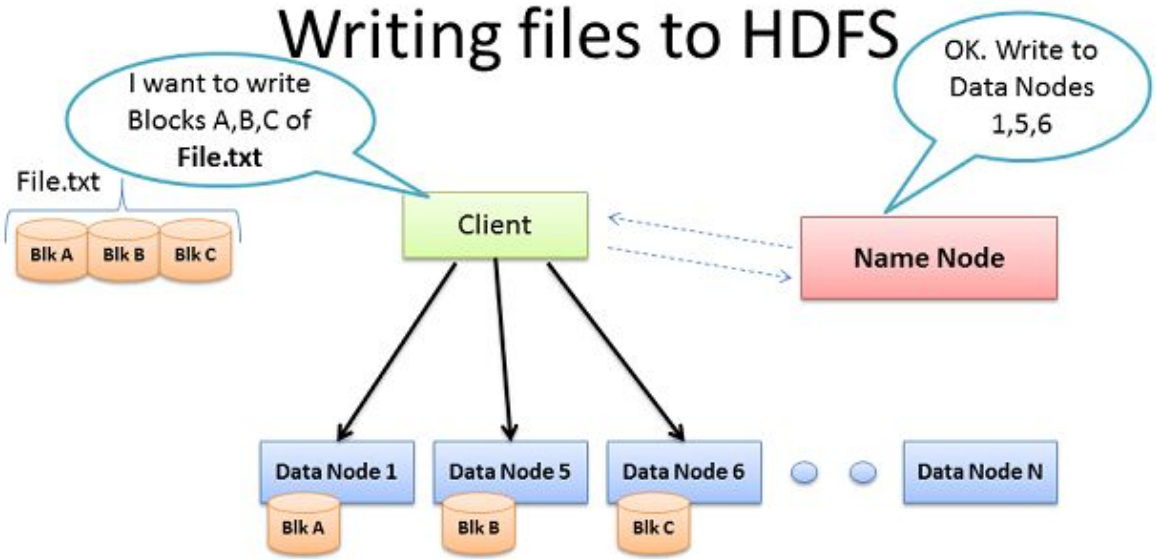
Sample Scenario:

How many times did our customers type the word "**Refund**" into emails sent to customer service?

Huge file containing all emails sent to customer service



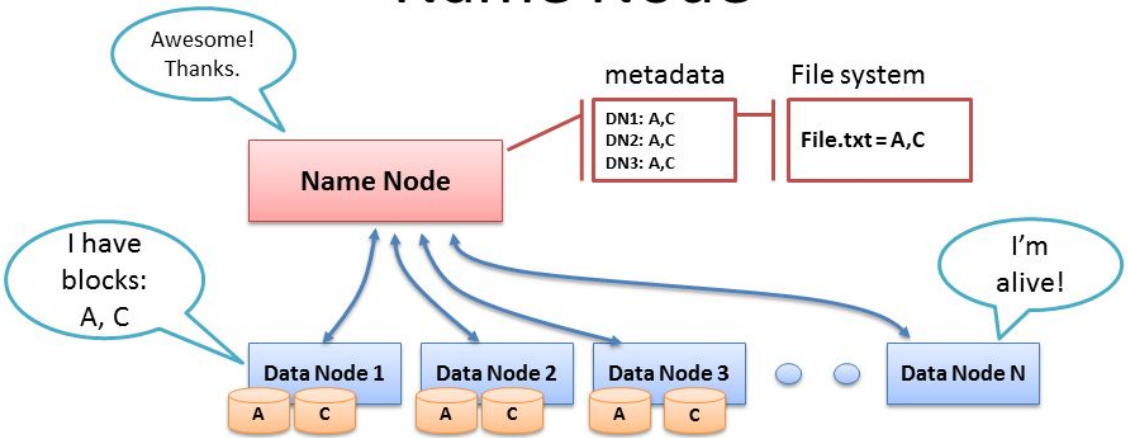
Apache Hadoop



- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

Apache Hadoop

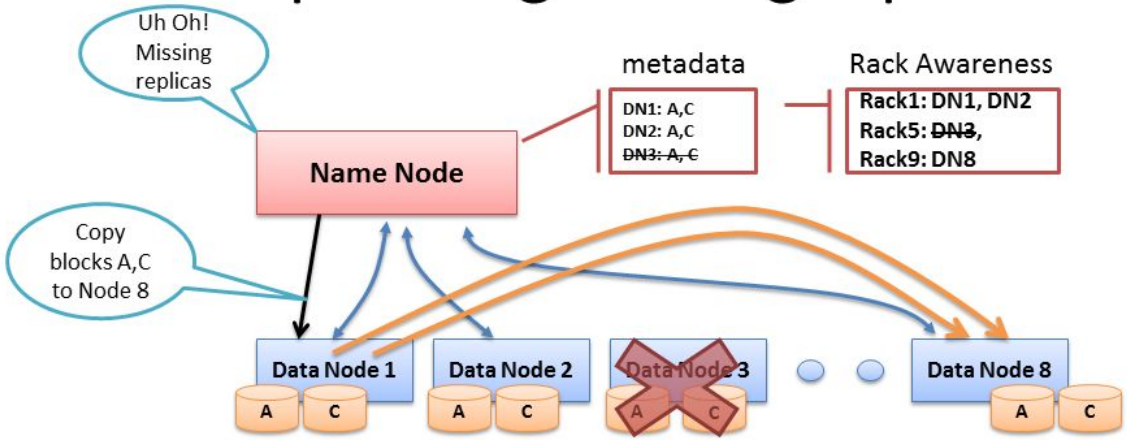
Name Node



- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds
- If Name Node is down, HDFS is down

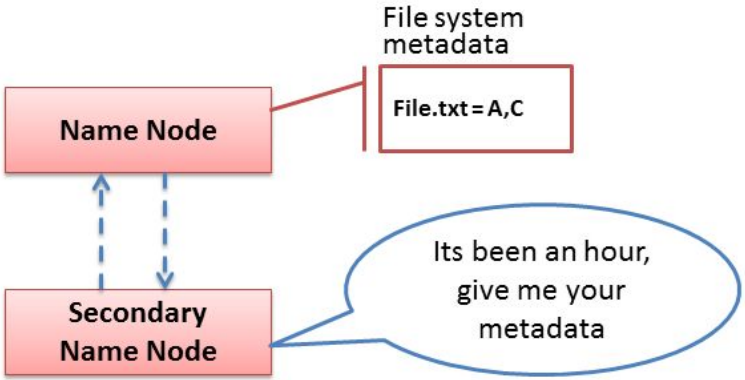
Apache Hadoop

Re-replicating missing replicas



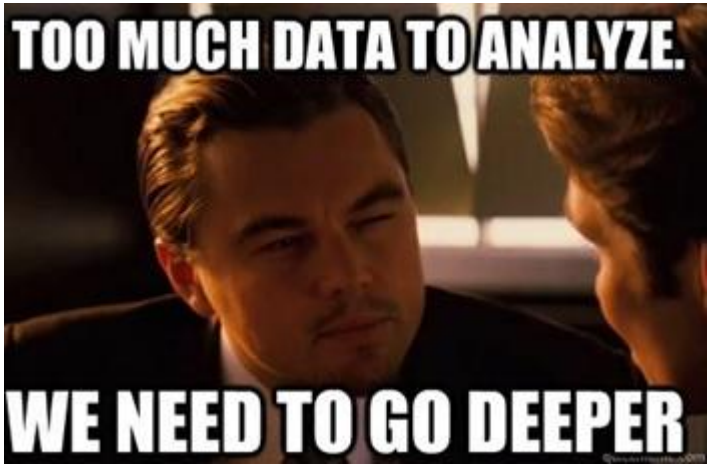
- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

Apache Hadoop
Secondary Name Node

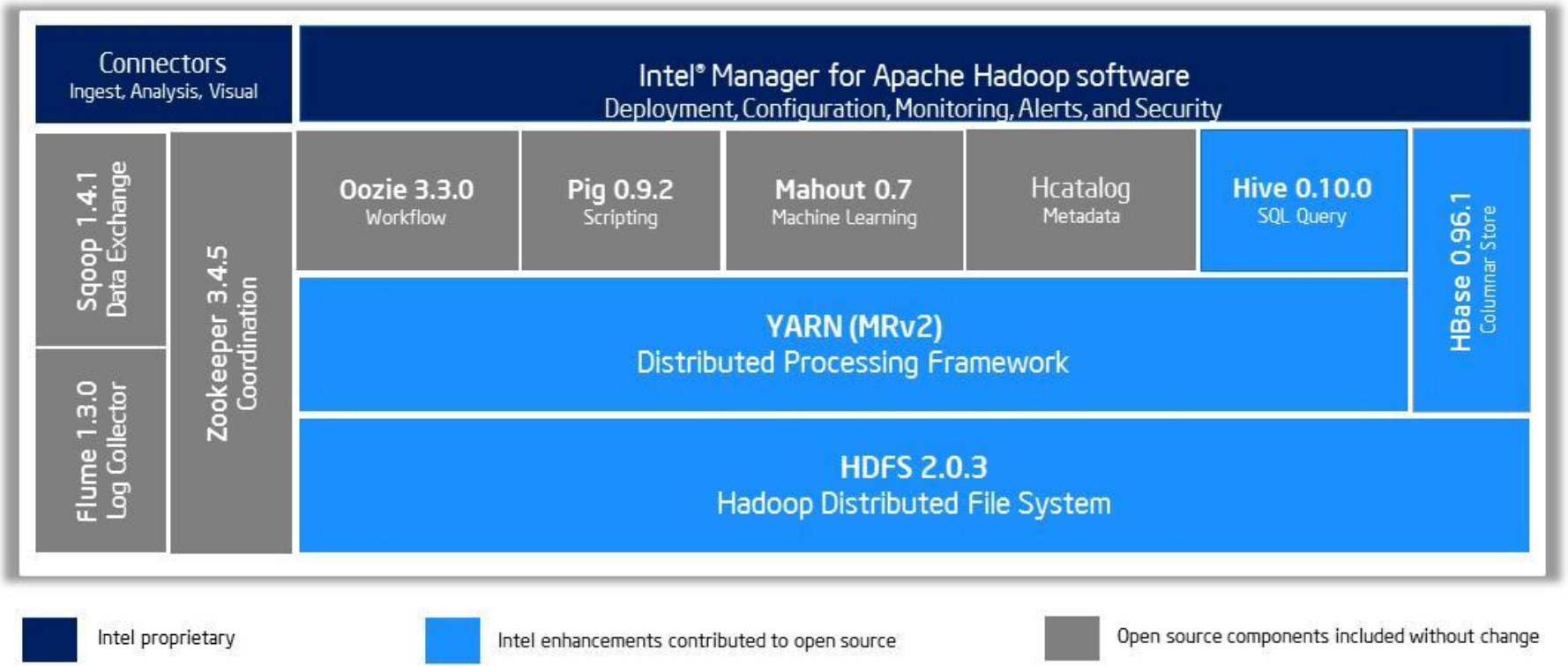


- Not a hot standby for the Name Node
- Connects to Name Node every hour*
- Housekeeping, backup of Name Node metadata
- Saved metadata can rebuild a failed Name Node

Apache Hadoop



Apache Hadoop



Apache Hadoop

GitHub This repository Search Explore Features Enterprise Pricing

facebookarchive / **hadoop-20** Watch 256

Facebook's Realtime Distributed FS based on Apache Hadoop 0.20-append

edit_generated_pom.py Sync github to internal FB hadoop code. 2 years ago

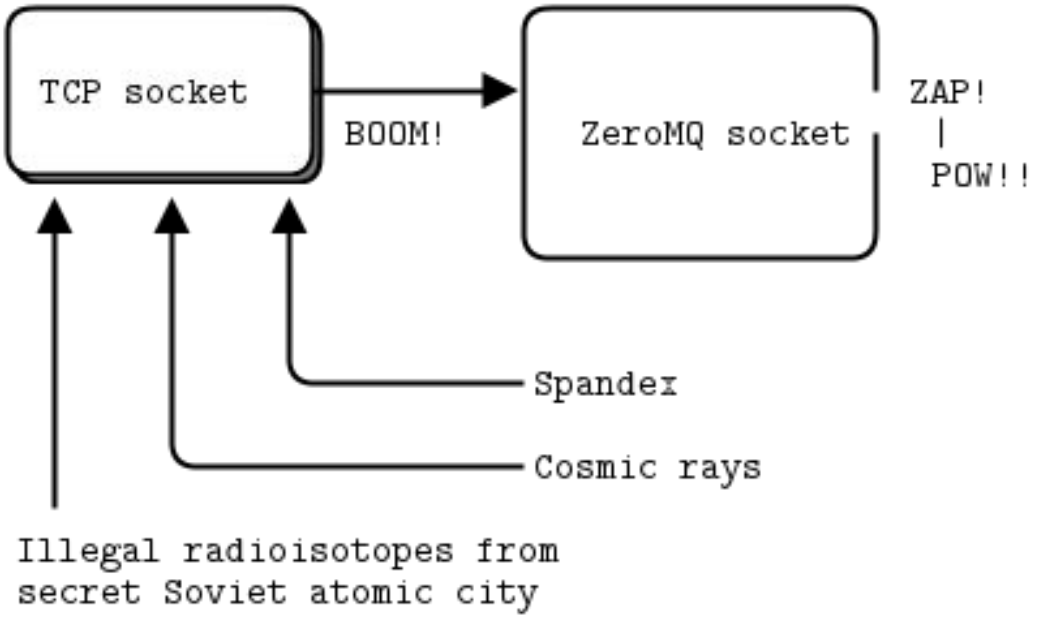
ZeroMQ

- Не является очередью сообщений в обычном понимании этого термина
- Является универсальным транспортным фреймворком
- Поддерживает большое количество языков программирования
- Обладает великолепной документацией
- Обладает великолепным кодом
- Проверена и используется большим количеством признанных лидеров индустрии

ZeroMQ

We took a normal TCP socket, injected it with a mix of radioactive isotopes stolen from a secret Soviet atomic research project, bombarded it with 1950-era cosmic rays, and put it into the hands of a drug-addled comic book author with a badly-disguised fetish for bulging muscles clad in spandex. Yes, ZeroMQ sockets are the world-saving superheroes of the networking world.

ZeroMQ


TCP socket
BOOM!
ZeroMQ socket
ZAP!
|
POW!!
Spandex
Cosmic rays
Illegal radioisotopes from
secret Soviet atomic city

ZeroMQ

- Различные типы сокетов (соединения 1-1, 1-много, соединения с/без гарантии доставки, round-robin балансировка и тд)
- Произвольный формат передаваемых данных между сокетами (возможен любой протокол, от бинарного до текстового)
- Необходимость чуть больше попрограммировать
- Большая гибкость получаемой системы

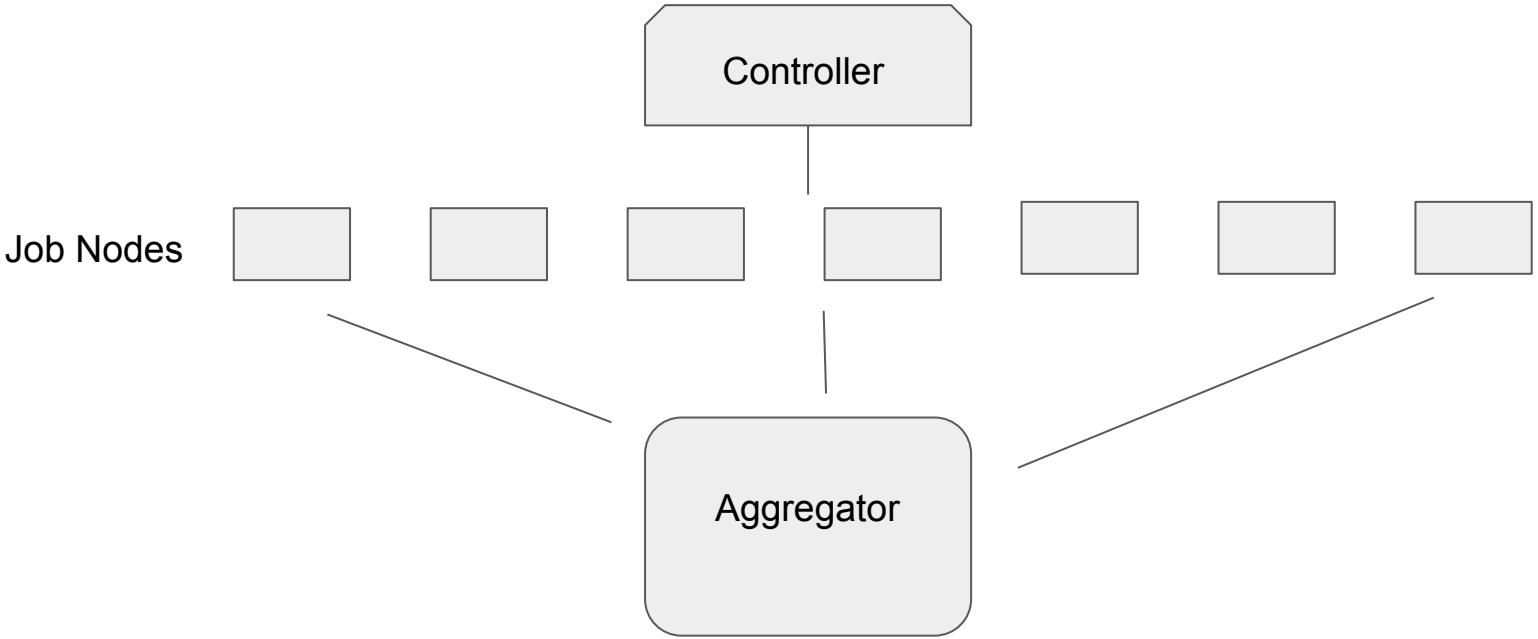
Обязательные вопросы

- В систему прилетает Task
- Task разбивается на маленькие Job
- Каждая Job мапится на соответствующего воркера
- Каждый воркер выполнения Job отсылает результаты в аггрегатор
- Аггрегатор дожидается выполнения всех Job и отдает итоговый результат

Обязательные вопросы

- Что делать если выполнение одной из Job по какой-либо причине сорвалось?
- Как определить, что выполнение сорвалось?
- Как долго ждать аггрегатору завершения выполнения всех Job?
- Как оценивать время выполнения всей Task в зависимости от доступных свободных Job-нод?
- ….

Базовая архитектура




That's all Folks!
SUPERB wallpapers