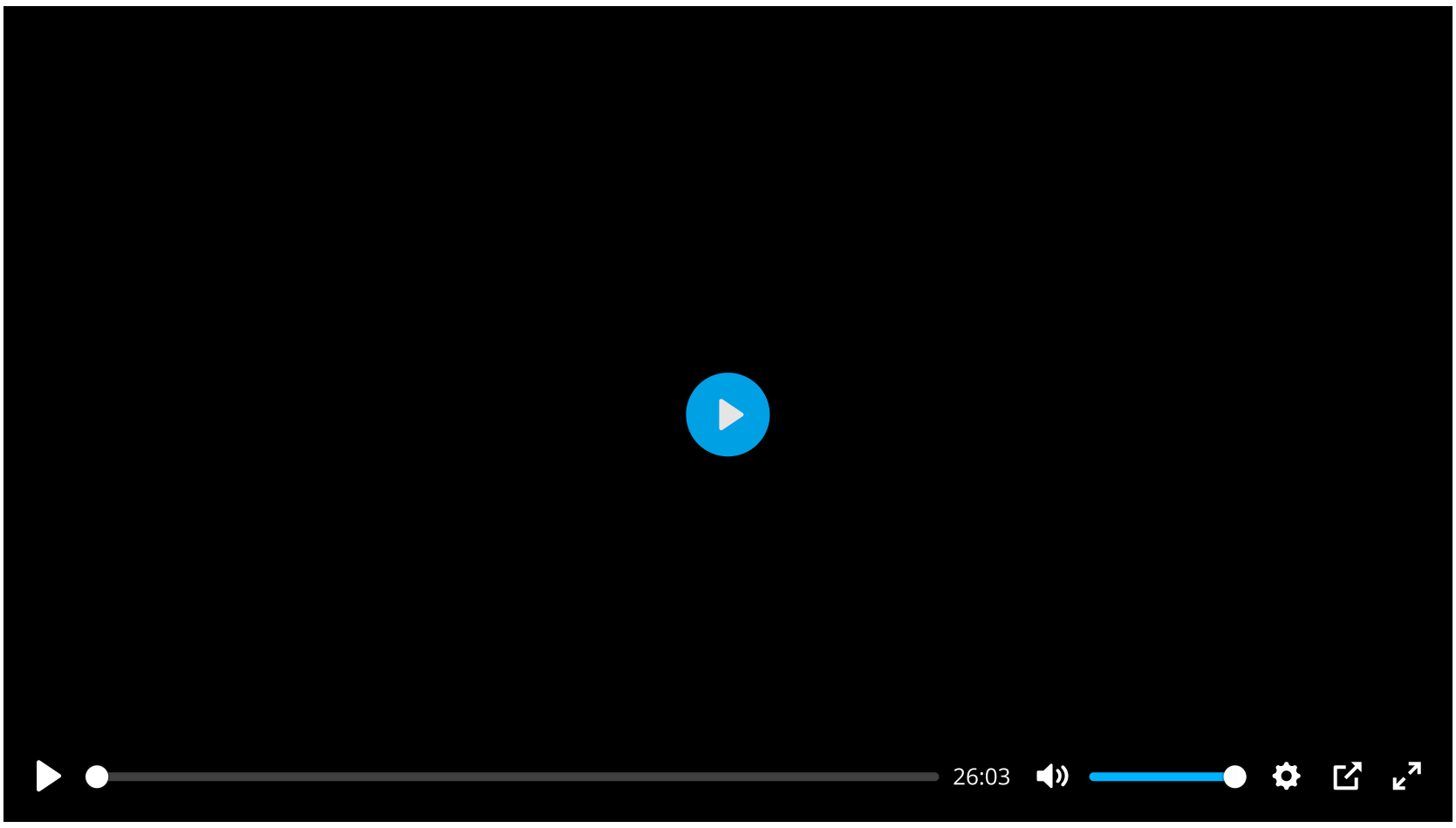
3.2 Экосистема Hadoop – краткий обзор и возможности



Текстовая расшифровка видео:

КАК ДАННЫЕ ХРАНЯТСЯ В HADOOP

План:

- Архитектура HDFS;
- NameNode, SNameNode, DataNode;
- Чуть больше про устройство HDFS;
- Хранение данных колоночно;
- Кодируемые данные – Parquet, Avro, Thrift;
- Потоковые данные – Kafka, Pulsar.

Архитектура HDFS

В Hadoop для хранения данных чаще всего используется распределенная файловая система HDFS (Hadoop Distributed File System). Архитектура HDFS использует практически те же идеи, лежащие в основе Google File System.

Ознакомиться со статьей о GFS вы можете, перейдя по ссылке:

https://research.google.com/archive/gfs-sosp2003.pdf

Плюс:

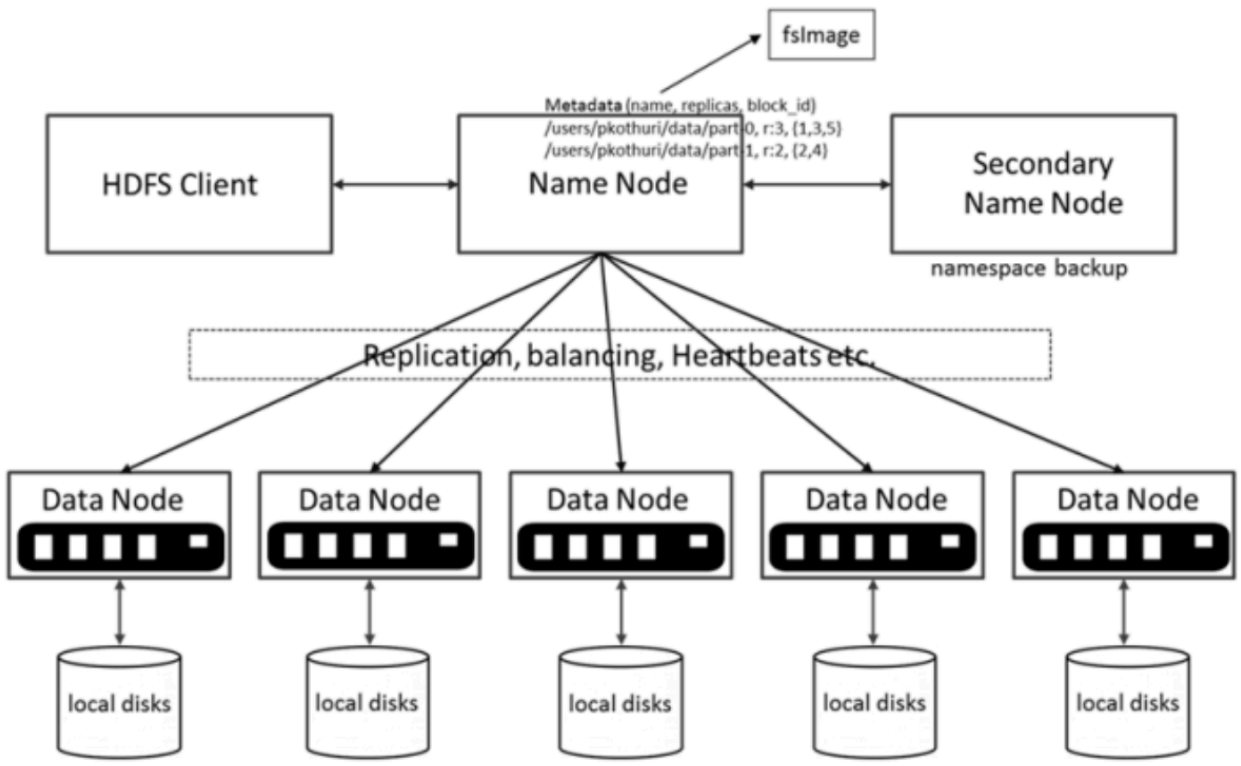
- Дает возможность использовать преимущества Data locality.

Минусы:

- Все упирается в одну ноду с метаданными;
- Не поддерживает совместимые драйвера файловых систем.

NameNode, SNameNode, DataNode

Пример того, как устроен HDFS:



Для хранения и вычисления данных логично запускать воркеры на тех машинах, где эти данные и лежат. Это позволит избежать проблем с распределенными системами. В ноде удобнее использовать собственную память, нежели прибегать к сторонним вариантам.

Благодаря тому, что информация о блоках и их местоположении хранится в NameNode, мы лучше понимаем, где запускать обработчики для минимизации копирования, достав их при помощи фреймворка.

В современном деплое HDFS содержится Secondary Namenode.

Важно: Secondary Namenode не является зеркалом NameNode! Secondary Namenode – дополнительное хранилище метаданных.

Datanode – хранилище нодов.

На NameNode запущен API, с которым мы можем взаимодействовать нашим клиентом с HDFS (кодом или консольными командами).

Чуть больше про устройство HDFS

- Файлы нарезаются на блоки размером 128Мб (по умолчанию). Не стоит хранить мелкие файлы, потому что блок все равно будет 128Мб.
- Каждый блок реплицируется столько раз, сколько задано в replication factor (по умолчанию 3). Это делается как с целью отказоустойчивости, так и с целью более быстрых вычислений.
- HDFS имеет достаточное количество встроенных механизмов. Он поддерживает Rack Awareness, имеет Replica Placement Policy. Replica Placement Policy.
- HDFS плохо подходит для хранения мелких файлов (много горизонтальной пересылки метаданных).

Хранение данных колоночно

Существует несколько баз данных, которые в качестве хранилища под собой используют HDFS. Одна из известных баз данных – **Apache Hbase**.

Основные положения:

1) NoSQL – база данных, которая работает поверх HDFS, используя свой формат HFile.

2) Apache Hbase построен по тому же принципу, что и Google Big Table (облачная база от Google, которая построена почти так же, как Hbase).

3) Таблицы не имеют схемы.

4) Можно задавать column families для более эффективной кластеризации.

5) Hbase позволяет версионировать записи (значения могут версионироваться и физически сортироваться).

Для работы с Hbase используется не SQL, а собственный механизм, с которым можно взаимодействовать. Даже поверх Hbase существуют еще более высокоуровневые решения для того, чтобы доставать из него данные.

Кодируемые данные – Parquet, Avro, Thrift

В Hadoop-проектах существуют несколько форматов хранения данных, которые учитывают размер блоков и нарезают файлы для более эффективного процессинга:

Parquet – один из самых популярных форматов. Служит для хранения данных at rest в колоночном формате для запросов. У него есть «брат-близнец» – ORC.

Apache iceberg – более специфичный формат экосистемы Hadoop. Подходит для очень больших данных и файлов.

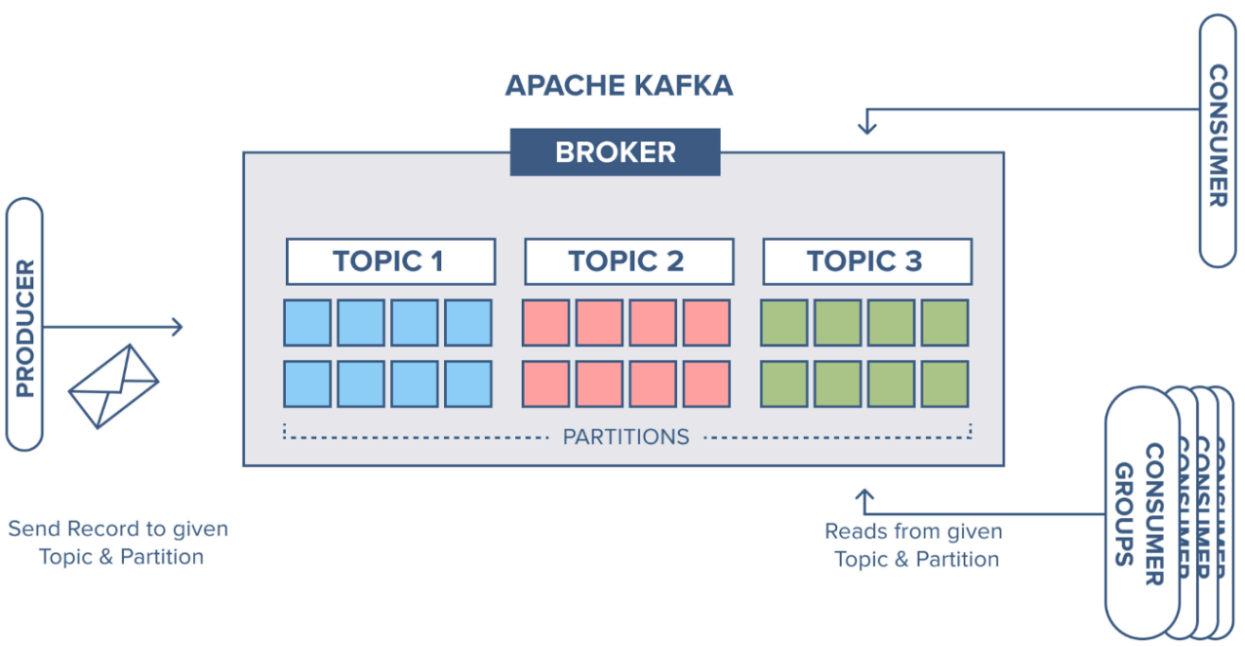
Avro – формат, предназначенный для пересылки данных on wire в JSON-совместимом формате.

Thrift – формат, предназначенный для выполнения remote procedure call (реализует протокол запуска задачи из одного сервиса в другой).

Потоковые данные – Kafka, Pulsar

Остановим внимание на **Apache Kafka** и **Apache Pulsar**.

Рассмотрим схему, относящуюся к Apache Kafka:



По схожести можно провести параллель между Apache Kafka и HDFS. Если HDFS все файлы/данные хранит на диске, то в случае Apache Kafka и Apache Pulsar используется труба, которая хранит в себе данные за определенный и конечный промежуток времени, где мы можем писать данные вначале и выкидывать их с конца. Все данные, которые мы записываем, делятся на кусочки, реплицируются и расходятся по кластеру.

Основное отличие: В случае **Kafk'и** и **Pulsar'а** система старается максимально сохранить данные в оперативной памяти, чтобы была возможность работать с ними в режиме реального времени. В HDFS же данные пишутся на диск.

Также следует упомянуть о наличии Partitions (записанные на разные машины куски данных) и topic'ов (маленькие ярлыки с именем тоннеля, куда мы отправляем данные).

Сравним Apache Kafka и Apache Pulsar:

Apache Kafka:

- Kafka Streams умеет мало;
- Брокеры – stateful;
- Умеет компактизировать топики при долгосрочном хранении.

Apache Pulsar:

- Делает базовые расчеты на данных самостоятельно;
- Брокеры – stateless;
- Pulsar поддерживает гео-репликацию.

Как вам урок? 😠 😐 🔥

Изучил, далее ›