**1. Apache Hadoop. Распределенная ФС HDFS, менеджер ресурсов YARN, парадигма MapReduce.**

**Hadoop**– это проект с открытым исходным кодом, находящийся под управлением Apache Software Foundation. Hadoop используется для надежных, масштабируемых и распределенных вычислений, но может также применяться и как хранилище файлов общего назначения, способное вместить петабайты данных. Многие компании используют Hadoop в исследовательских и производственных целях.

Hadoop состоит из двух ключевых компонентов:

- Распределенная файловая система Hadoop (HDFS), которая отвечает за хранение данных на кластере Hadoop;

- Система MapReduce, предназначенная для вычислений и обработки больших объемов данных на кластере.

**HDFS** (*Hadoop Distributed File System*) — файловая система, предназначенная для хранения файлов больших размеров, поблочно распределённых между узлами вычислительного кластера. Все блоки в HDFS (кроме последнего блока файла) имеют одинаковый размер, и каждый блок может быть размещён на нескольких узлах, размер блока и коэффициент репликации (количество узлов, на которых должен быть размещён каждый блок) определяются в настройках на уровне файла. Благодаря репликации обеспечивается устойчивость распределённой системы к отказам отдельных узлов. Файлы в HDFS могут быть записаны лишь однажды (модификация не поддерживается), а запись в файл в одно время может вести только один процесс. Организация файлов в пространстве имён — традиционная иерархическая: есть корневой каталог, поддерживается вложение каталогов, в одном каталоге могут располагаться и файлы, и другие каталоги.

Развёртывание экземпляра HDFS предусматривает наличие центрального *узла имён* (*name node*), хранящего метаданные файловой системы и метаинформацию о распределении блоков, и серии *узлов данных* ( *data node*), непосредственно хранящих блоки файлов. Узел имён отвечает за обработку операций уровня файлов и каталогов — открытие и закрытие файлов, манипуляция с каталогами, узлы данных непосредственно отрабатывают операции по записи и чтению данных. Узел имён и узлы данных снабжаются веб-серверами, отображающими текущий статус узлов и позволяющими просматривать содержимое файловой системы. Административные функции доступны из интерфейса командной строки.



**YARN** (*Yet Another Resource Negotiator*) — модуль, отвечающий за управление ресурсами кластеров и планирование заданий. В YARN функционирует логически самостоятельный демон — планировщик ресурсов (*ResourceManager*), абстрагирующий все вычислительные ресурсы кластера и управляющий их предоставлением приложениям распределённой обработки. Работать под управлением YARN могут как MapReduce-программы, так и любые другие распределённые приложения, поддерживающие соответствующие программные интерфейсы; YARN обеспечивает возможность параллельного выполнения нескольких различных задач в рамках кластера и их изоляцию.



**MapReduce** — модель распределённых вычислений, представленная компанией Google, используемая для параллельных вычислений над очень большими наборами данных.

MapReduce — это фреймворк для вычисления некоторых наборов распределенных задач с использованием большого количества компьютеров (называемых «нодами»), образующих кластер.

Работа MapReduce состоит из двух шагов: Map и Reduce, названных так по аналогии с одноименными функциями высшего порядка, map и reduce.

На Map-шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом — master node) получает входные данные задачи, разделяет их на части и передает другим компьютерам (рабочим узлам — worker node) для предварительной обработки.

На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат — решение задачи, которая изначально формулировалась.

Преимущество MapReduce заключается в том, что он позволяет распределенно производить операции предварительной обработки и свертки. Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно. Аналогично, множество рабочих узлов может осуществлять свертку — для этого необходимо только чтобы все результаты предварительной обработки с одним конкретным значением ключа обрабатывались одним рабочим узлом в один момент времени. Параллелизм также дает некоторые возможности восстановления после частичных сбоев серверов: если в рабочем узле, производящем операцию предварительной обработки или свертки, возникает сбой, то его работа может быть передана другому рабочему узлу.

**2. Apache HBase.**

**HBase** - это колоночная база данных, расположенная поверх HDFS, которая способна вместить огромные массивы данных (от гигабайтов до петабайтов данных). HBase используется тогда, когда нужен произвольный доступ к данным, хранящимся в HDFS, в реальном времени и с правом записи и чтения.

Модель данных Hbase можно запомнить как соответствие ключ значение:

**<table, RowKey, Column Family, Column, timestamp> -> Value**

****

Операции: **put, get, scan** (читать записи по очереди)**, delete** (физического удаления до major compaction не будет)

Hbase для своей работы использует два основных процесса:   
**1. Region Server**— обслуживает один или несколько **регионов.**Регион — это диапазон записей соответствующих определенному диапазону подряд идущих RowKey. Каждый регион содержит:

* **Persistent Storage**— основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате **HFile**. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle.
* **MemStore** — буфер на запись. Так как данные хранятся в HFile d отсортированном порядке — обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении MemStore до некоторого критического значения данные записываются в новый HFile.
* **BlockCache** — кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.
* **Write Ahead Log(WAL).**Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не произошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.

**2. Master Server** — главный сервер в кластере hbase. Master управляет распределением регионов по Region Server’ам, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.   
При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.   
Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется **compaction.** Compaction’ы бывают двух видов:

* **Minor Compaction.**Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.
* **Major Compaction.**Запускается руками или по наступлению срабатыванию определенных триггеров(например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера. **Major Compaction**’ы лучше делать во время когда нагрузка на кластер небольшая. Во время Major Compaction также происходит физическое удаление данных, ране помеченных меткой tombstone.

**3. Apache Pig.**

**Pig это:**

* язык Pig Latin
* Интерактивная консоль
* Встроенные функции агрегации
* Поддержка пользовательских функций (UDF)
* Данные — в виде структур (Tuple, Bag)

**Pig умеет работать:**

* с текстовыми файлами (можно задать разграничительный символ)
* с сжатыми текстовыми файлами (Gzip, Bzip)
* с массивами, словарями, объединениями (union)
* имеет огромное количество встроенных функций для работы с: датами, строками, структурами
* с математическими функциями (округление, логарифмы, корни, тригонометрия)
* с функциями агрегации (sum, min, max, avg...)
* Если всего перечисленного выше не хватило, то можно использовать кастомные функции (jython, java)

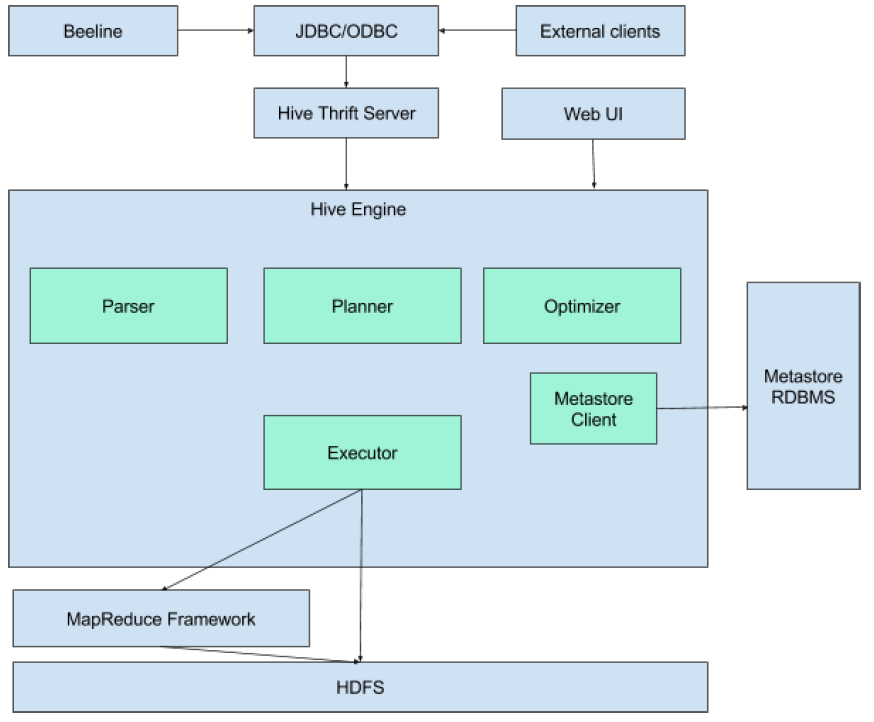
Как видите, Pig умеет всё то же, что и Hive. Отличие лишь в представлении данных и в языке. Но именно это отличие выводит работу с Pig совершенно на другой уровень.  
Данный фреймфорк работает со специальными структурами данных — Tuple и Bag.

* **Tuple** — упорядоченный набор полей. Структура, к полям которой можно обращаться по индексу и/или имени.
* **Bag** — коллекция (множество) Tuple.

**4. Apache Hive.**

Hive представляет из себя движок, который превращает SQL-запросы в цепочки MapReduce задач.

* SQL-подобный язык HQL;
* интерактивная консоль;
* встроенные функции агрегации;
* поддержка пользовательских функций (UDF);
* данные — как таблица.



* база данных
* таблица
* партиция
* бакет

База данных представляет аналог базы данных в реляционных СУБД - пространство имен, содержащее таблицы.

Данные в таблицах Hive хранятся в виде обычных файлов на HDFS. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и другие форматы.

Для того чтобы избежать при запросах полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS.

В один бакет попадают строчки таблицы, для которых совпадает значение хэш-функции, вычисленное по определенной колонке.

**5. Apache Storm.**

Storm — это пример сложной системы обработки событий (Complex Event-Processing — CEP). CEP-системы обычно подразделяют на вычислительные и регистрирующие, и каждый из этих классов можно реализовать в Storm с помощью пользовательских алгоритмов. Например, CEP-системы можно использовать для выявления значимых событий в потоке событий, а затем принимать меры в режиме реального времени (тенденции в Twitter).

Storm реализует модель передачи данных, в которой данные непрерывно проходят сквозь цепь преобразований. Абстракция для передачи данных называется потоком (stream) и представляет собой бесконечную последовательность кортежей (tuples). Кортеж напоминает структуру, которая может содержать стандартные типы данных (массивы целых чисел, чисел с плавающей запятой и

байтов) или типы, определяемые пользователем, с некоторым дополнительным кодом сериализации.

Каждый поток помечается уникальным идентификатором, который можно использовать для построения топологии источников и приемников данных. Источниками потоков служат воронки (spouts), передающие данные из внешних источников в топологию Storm. Приемники (или объекты, производящие преобразования) называются ситами (bolts). Они выполняют отдельные преобразования потока и всю обработку в топологии Storm.

**6. Apache Spark.**

Apache Spark — программный каркас с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. В отличие от классического обработчика из ядра Hadoop, реализующего двухуровневую концепцию MapReduce с дисковым хранилищем, использует специализированные примитивы для рекурентной обработки в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы для некоторых классов задач, в частности, возможность многократного доступа к загруженным в память пользовательским данным делает библиотеку привлекательной для алгоритмов машинного обучения. Проект предоставляет программные интерфейсы для языков Java, Scala, Python, R.

В Spark вводится концепция RDD (устойчивый распределенный набор данных) — неизменяемая отказоустойчивая распределенная коллекция объектов, которые можно обрабатывать параллельно. В RDD могут содержаться объекты любых типов; RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы (driver program).

В RDD поддерживаются операции двух типов:

* Трансформации — это операции (например, отображение, фильтрация, объединение и т.д.), совершаемые над RDD; результатом трансформации становится новый RDD, содержащий ее результат.
* Действия — это операции (например, редукция, подсчет и т.д.), возвращающие значение, получаемое в результате некоторых вычислений в RDD.

Жизненный цикл программы:

* локальный запуск драйвера, подготовка конвейера
* общение с мастером, рассылка кода конвейера и плана
* выполнения кластеру
* распределённое выполнение конвейера
* отсылка результатов обратно драйверу

RDD распределяется по кластеру в виде совокупности секций, каждая из которых включает подмножество данных. Секции в Spark являются определяющим фактором параллелизма. Объекты в секции обрабатываются последовательно, а несколько секций обрабатывается параллельно.

**7. Apache Giraph.**

Giraph — это фреймворк для итеративной обработки больших графов, который работает поверх системы Hadoop.   
Особенность Giraph — vertex-centric-модель.

Вершина в этой модели может обмениваться сообщениями с другими вершинами между итерациями. Параллелизацией и масштабированием занимается сам Giraph.

Обработка графа в Giraph выглядит следующим образом: процесс разбит на итерации, которые называются суперстепами (supersteps). На каждом суперстепе вершина выполняет необходимую программу и, если надо, может разослать сообщения другим вершинам. На следующей итерации вершина получает сообщения, выполняет программу, рассылает сообщения и т. д. После завершения всех суперстепов вы получите результирующий граф.  
Giraph поддерживает большое количество возможностей взаимодействия с графом, в том числе создание/удаление вершин, создание/удаление ребер, возможность переопределения формата, в котором задан граф или выбор из существующих, управление загрузкой с диска и выгрузкой частей графа на диск во время работы с ним и т. д.

Пользовательская функция определяет поведение для отдельной вершины V и отдельного супершага S. Функция может считывать сообщения, отправляемые вершине V на супершаге S-1, посылать сообщения другим вершинам (они будут получены на супершаге S+1), а также изменять состояние вершины V и исходящих из нее ребер.

Первый шаг при программировании Giraph-приложений — определение входных данных. Это:

* идентификатор вершины
* опционально — значение вершины
* кортежи вида: <идентификатор вершины назначения, вес ребра (опционально)>

Затем необходимо определить формат входных данных, или использовать уже имеющийся.