АРХИТЕКТУРА HDFS

HDFS

Hadoop Distributed File System (HDFS) – распределенная файловая система Hadoop, одна из основных компонент Apache Hadoop

HDFS предназначена для хранения файлов больших размеров в распределенной среде, т.е. в рамках кластера из нескольких узлов

Как и любая файловая система, HDFS представляет из себя иерархию каталогов с вложенными в них подкаталогами и файлами

HDFS может быть использован для запуска MapReduce-задач, а также как распределенная файловая система общего назначения, обеспечивающая работу распределенных СУБД (HBase) и масштабируемых систем машинного обучения

ЦЕЛЕСООБРАЗНОСТЬ

Выход из строя сервера – ситуация, которая встречается в процессе работы с некоторой ненулевой вероятностью. При работе с большим объемом данных (размер одного файла – гига- и терабайты) часто приходится использовать распределенные системы

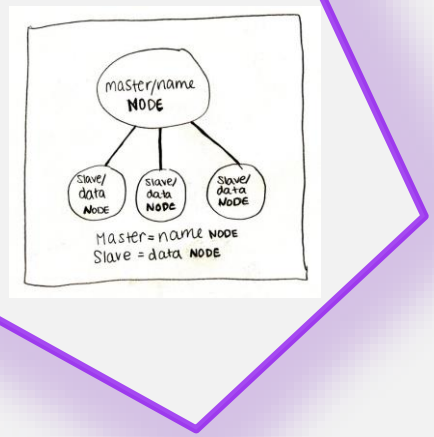
Один кластер для работы с данными может содержать сотни или тысячи серверов. Таким образом, следует принять за данность тот факт, что определенная часть из них всегда будет нерабочей

Ключевые архитектурные задачи, которые решает HDFS – высокая пропускная способность, высокая отказоустойчивость, быстрое реагирование на возникающие неполадки

HDFS: БАЗОВЫЕ ПРИНЦИПЫ УСТРОЙСТВА

HDFS использует несколько серверов **(nodes)**

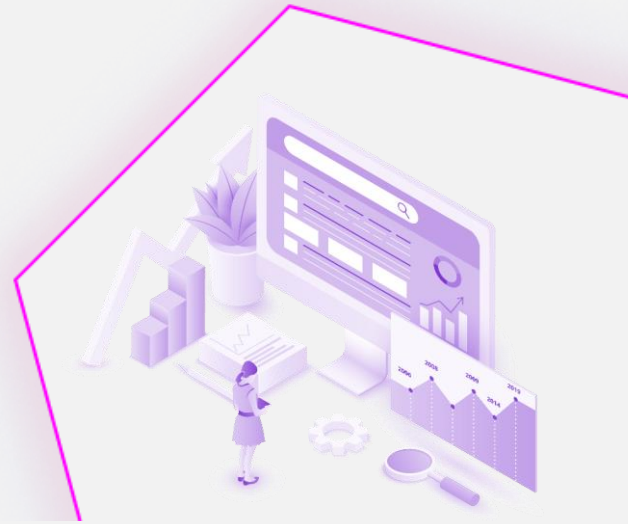
Большие файлы разделяются на куски – блоки (как правило, 64 или 128 Мб). Блоки могут подвергаться репликации: на каждый блок приходится несколько копий, которые хранятся на разных серверах.



Один из серверов называется **NameNode**, его задача состоит в том, чтобы хранить информацию о местонахождении блоков файлов

Остальные сервера называются **DataNode**, они отвечают непосредственно за запись и чтение данных, получая инструкции от **NameNode**

NAMENODE

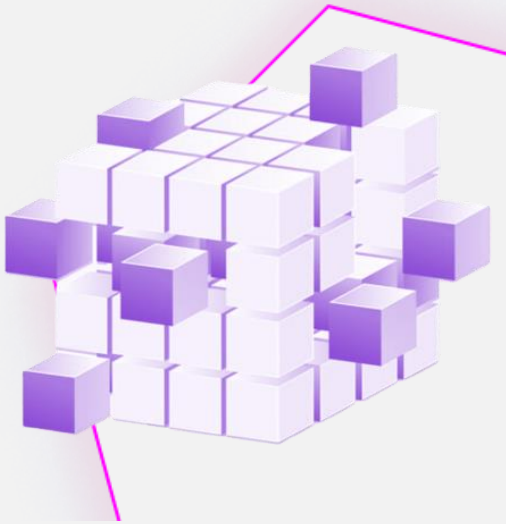


NameNode – управляющий узел, узел имен или сервер имен – отдельный, единственный в кластере. Содержит программный код для управления пространством имен файловой системы, хранящий дерево файлов, а также мета-данные файлов и каталогов

NameNode отвечает за открытие и закрытие файлов, создание и удаление каталогов, управление доступом со стороны внешних клиентов, соответствие между файлами, блоками и их репликами (копиями на узлах)

Раскрывает для всех желающих расположение блоков данных на машинах кластера

SECONDARY NAMENODE

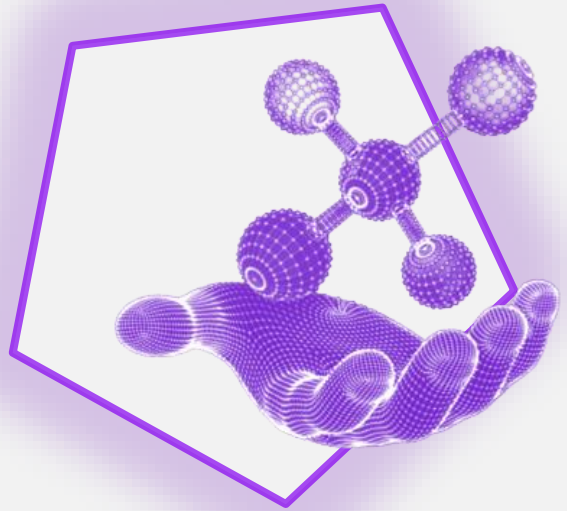


Secondary NameNode – вторичный узел имен – отдельный, единственный в кластере

Необходим для быстрого ручного восстановления NameNode в случае его выхода из строя

Принцип работы: Образ HDFS (то есть, необходимый набор ПО и информация о файловой системе) и лог транзакций произведенных операций копируется во временную папку, применяет изменения, накопленные в логе транзакций к образу HDFS, после чего записывает обновленный образ на NameNode и очищает лог транзакций

DATANODE



DataNode – узел или сервер данных – один из множества серверов кластера, отвечающий за файловые операции и непосредственно работу с блоками

- Выполняет запись, чтение данных, создание, удаление и репликацию блоков
- Получает команды от NameNode на манипуляции с блоками и репликами
- Обрабатывает запросы от клиентов файловой системы HDFS на чтение и запись, минуя NameNode
- Отправляет сообщения о состоянии, называемые heartbeats (сердцебиения)

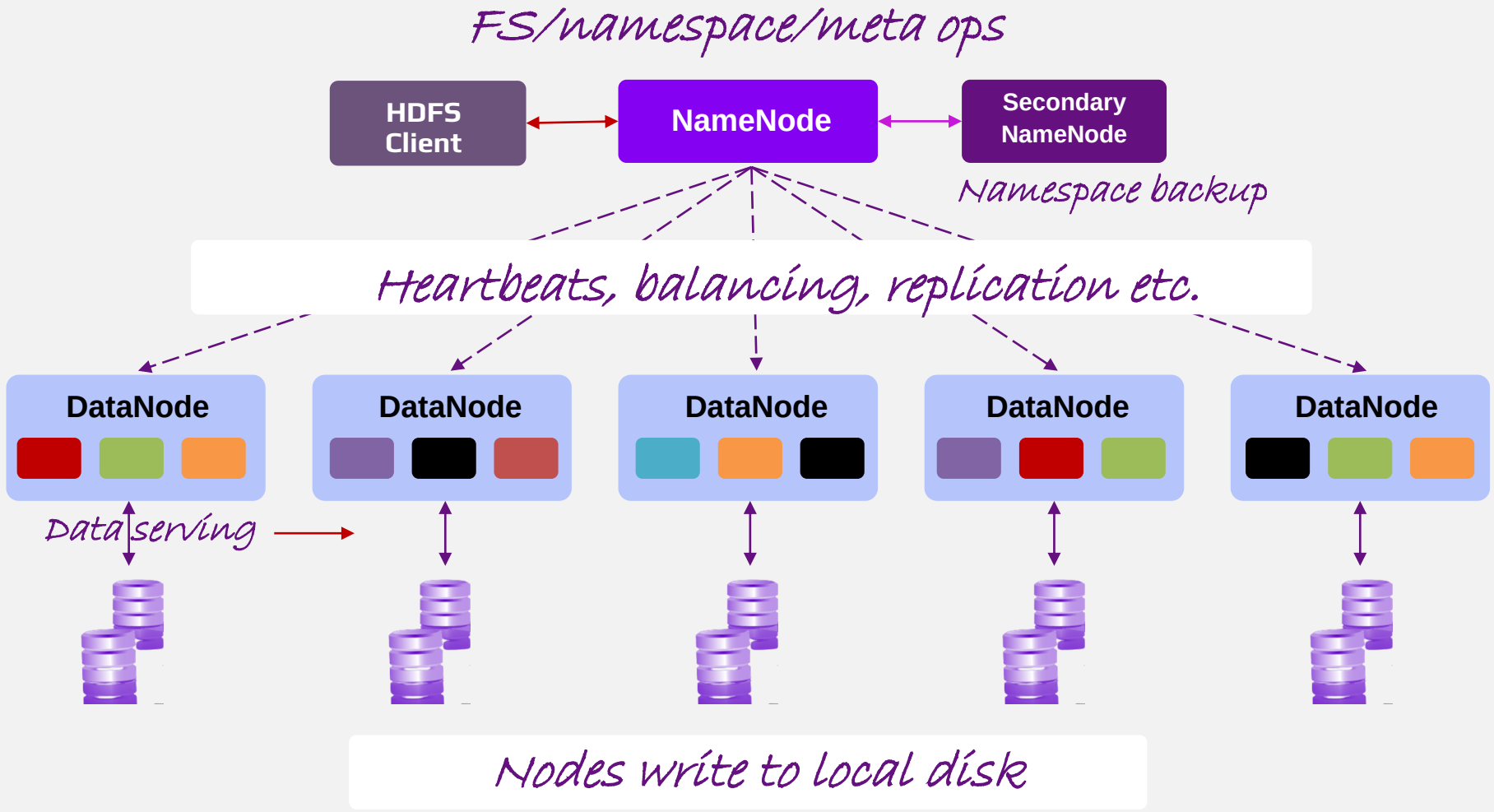
CLIENT

Client – клиент – пользователь или приложение, взаимодействующий через интерфейс с распределенной файловой системой

Клиент может иметь различные права на создание, удаление, чтение, запись, переименование или перемещение файлов и каталогов

При создании файла клиент может задать размер блока (по умолчанию – 64 Мб), а также количество создаваемых реплик (по умолчанию это значение равно 3, то есть каждый блок присутствует в файловой системе в виде трех копий)

HDFS: ПОДРОБНАЯ СХЕМА АРХИТЕКТУРЫ



HDFS: ПРЕИМУЩЕСТВА

1. Ориентация на недорогие сервера: ненадежность машин компенсируется высокой отказоустойчивостью самой файловой системы (за счет репликации данных)

2. Асинхронная репликация: блоки файлов реплицируются прямо во время загрузки, поэтому выход из строя отдельных узлов не повлечет пропажу данных

3. Возможность для клиента считывать и писать файлы напрямую в HDFS

4. Принцип «write-once, read-many»: запись в файл в любой момент времени доступен максимум одному процессу, конфликты множественной записи исключены

HDFS: НЕДОСТАТКИ

1. Отказ сервера имен влечет отказ всей системы
2. Отсутствие полноценной возможности изменения блоков данных
3. Отсутствие полноценной репликации Secondary NameNode

СТРУКТУРА ФАЙЛОВ

Файлы в HDFS разделены на блоки, размер которых может быть задан вручную. По умолчанию размер блока равен 64 Мб

Приложения, работающие с HDFS, записывают данные только один раз, но читать их могут многократно, и требуют скоростей чтения, близких к стриминговым

Стратегия репликации реализована следующим образом: клиент получает от NameNode список узлов данных (DataNodes) в количестве, равном фактору репликации (т.е. количеству копий на каждый блок). Первый узел данных получает данные небольшими порциями (4 кб), записывает их в локальную файловую систему, после чего посылает эти данные следующему узлу. Второй узел делает то же самое, и так далее. Таким образом, узел данных может одновременно получать данные и отправлять их на следующий узел для репликации

API

Доступ к HDFS может быть осуществлен множеством различных способов.
Типичный способ доступа к данным – Java API (Application Programming Interface).
Ниже мы рассмотрим типичные команды для работы с оболочкой HDFS.
В целом, команды очень похожи на команды Linux Shell

hdfs dfs -ls / Просмотр файлов или папок по корневому пути
hdfs dfs -mkdir -p /AAA/BBB для рекурсивного создания папок, без -p нерекурсивно
hdfs dfs -moveFromLocal <sourceDir> (Файл на локальном диске) <hdfsDir> (путь HDFS)
hdfs dfs -mv <sourceDir> (Исходные файлы HDFS) <destDir> (путь назначения HDFS)
hdfs dfs -put <localDir> hdfsDir загрузить файл Dao1HDFS
hdfs dfs -appendToFile a.txt b.txt /hello.txt объединить файлы в HDFS
hdfs dfs -cat <hdfsDir> Просмотр файлов HDFS
hdfs dfs -cp <HDFSSourceDir> <HDFSDestDir> копировать файлы
hdfs dfs -rm [-r] Удалить файлы и папки (-r означает рекурсивное удаление)
hdfs dfs -chomd -R 777 /xxx изменить права доступа к файлам

OBJECT STORAGE VS HDFS

Object Storage (например, AWS S3) – не является файловой системой, все данные хранятся в виде объектов (Storage Entities) с ключом, значением и версией

Свойство	HDFS	Object Storage
Архитектура	NameNode, поддерживающая пространство имен, несколько DataNodes для процессинга данных	Объект состоит из данных и идентификатора. Все метаданные располагаются вне хранилища
Внутренняя структура	Традиционная (иерархическая). Пользователь может создавать, перемещать, переименовывать файлы	Нет структуры директорий, все хранится в плоском адресном пространстве. Поддерживает только две операции: PUT (создание нового объекта и заполнение его данными) и GET (получение объекта по идентификатору)

OBJECT STORAGE VS HDFS

Свойство	HDFS	Object Storage
Согласованность	Клиенты видят содержимое файлов идентично ситуации, когда каждый файл представлен в виде одной копии	Создание, удаление и обновление объекта требует времени, прежде чем эти изменения будут видны всем пользователям. Старые копии файлов могут существовать неопределенное время
Атомарность	Операции переименования и удаления атомарны (то есть, выполняются только полностью для объекта)	Изменения могут быть не атомарны. Если операция прервана в процессе работы, то объект отражает частичные изменения

СПАСИБО ЗА ВНИМАНИЕ