**ЛАБОРАТОРНАЯ РАБОТА №1**

**ОСНОВЫ HADOOP. УСТАНОВКА HADOOP. ОСНОВНЫЕ КОМАНДЫ ФАЙЛОВОЙ СИСТЕМЫ HDFS**

**Цель работы:** формирование практических навыков по установке и настройке кластера Hadoop и работе с файловой системой HDFS.

**Задачи:**

1. Изучить основы Hadoop.
2. Научиться устанавливать и конфигурировать Hadoop.
3. Изучить основные команды для работы с файловой системой HDFS.
4. Получить навыки написания программ для работы с HDFS.

Результатами работы являются:

− Настроенный Hadoop-кластер

− Программа, использующая HDFS API, для решения задачи согласно варианту задания

− Подготовленный отчет

# КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ,

# ИССЛЕДОВАНИЯ

Hadoop – это свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Эта основополагающая технология хранения и обработки больших данных (Big Data) является проектом верхнего уровня фонда Apache Software Foundation.

**Из чего состоит Hadoop: концептуальная архитектура**

Изначально проект разработан на Java в рамках вычислительной парадигмы MapReduce, когда приложение разделяется на большое количество одинаковых элементарных заданий, которые выполняются на распределенных компьютерах (узлах) кластера и сводятся в единый результат.

Проект состоит из основных 4-х модулей:

1. Hadoop Common – набор инфраструктурных программных библиотек и утилит, которые используются в других решениях и родственных проектах, в частности, для управления распределенными файлами и создания необходимой инфраструктуры;
2. HDFS – распределённая файловая система, Hadoop Distributed File System – технология хранения файлов на различных серверах данных (узлах, DataNodes), адреса которых находятся на специальном сервере имен (мастере, NameNode). За счет дублирования (репликации) информационных блоков, HDFS обеспечивает надежное хранение файлов больших размеров, поблочно распределённых между узлами вычислительного кластера;
3. YARN – система планирования заданий и управления кластером (Yet Another Resource Negotiator), которую также называют MapReduce 2.0 – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений. Фактически, YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и обработки данных;
4. Hadoop MapReduce – платформа программирования и выполнения распределённых MapReduce-вычислений, с использованием большого количества компьютеров (узлов, nodes), образующих кластер.

**Где и зачем используется Hadoop**

Выделяют несколько областей применения технологии:

* поисковые и контекстные механизмы высоконагруженных веб-сайтов и интернет-магазинов (Yahoo!, Facebook, Google, AliExpress, Ebay и т.д.), в т.ч. для аналитики поисковых запросов и пользовательских логов;
* хранение, сортировка огромных объемов данных и разбор содержимого чрезвычайно больших файлов;
* быстрая обработка графических данных, например, газета New York Times с помощью хадуп и Web-сервиса Amazon Elastic Compute Cloud всего за 36 часов преобразовала 4 терабайта изображений в PNG-формат размером по 800 КБ.

**Hadoop Distributed File System**

**HDFS** — распределенная файловая система Hadoop для хранения файлов больших размеров с возможностью потокового доступа к информации, поблочно распределённой по узлам вычислительного кластера, который может состоять из произвольного аппаратного обеспечения. Hadoop Distributed File System, как и любая файловая система – это иерархия каталогов с вложенными в них подкаталогами и файлами.

**Архитектура HDFS**

Кластер HDFS включает следующие компоненты:

1. Управляющий узел, узел имен или сервер имен (NameNode) – отдельный, единственный в кластере, сервер с программным кодом для управления пространством имен файловой системы, хранящий дерево файлов, а также мета-данные файлов и каталогов.
2. Secondary NameNode — вторичный узел имен, отдельный сервер, единственный в кластере, который копирует образ HDFS и лог транзакций операций с файловыми блоками во временную папку, применяет изменения, накопленные в логе транзакций к образу HDFS, а также записывает его на узел NameNode и очищает лог транзакций. Secondary NameNode необходим для быстрого ручного восстанавления NameNode в случае его выхода из строя.
3. Узел или сервер данных (DataNode, Node) – один их множества серверов кластера с программным кодом, отвечающим за файловые операции и работу с блоками данных. DataNode – обязательный компонент кластера HDFS, который отвечает за запись и чтение данных, выполнение команд от узла NameNode по созданию, удалению и репликации блоков, а также периодическую отправку сообщения о состоянии (heartbeats) и обработку запросов на чтение и запись, поступающих от клиентов файловой системы HDFS.
4. Клиент (client) – пользователь или приложение, взаимодействующий через специальный интерфейсс распределенной файловой системой. При наличии достаточных прав, клиенту разрешены следующие операции с файлами и каталогами: создание, удаление, чтение, запись, переименование и перемещение.

**Yet Another Resource Negotiator**

**YARN** – это система планирования заданий и управления кластером, которую также называют MapReduce 2.0 – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений. YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и аналитики больших данных.

**Архитектура и принципы работы**

YARN состоит из следующих компонентов:

* ResourceManager (RM) – менеджер ресурсов, которых отвечает за распределение ресурсов, необходимых для работы распределенных приложений, и наблюдение за узлами кластера, где эти приложения выполняются.
* ApplicationMaster (AM) – мастер приложения, ответственный за планирование его жизненного цикла, координацию и отслеживание статуса выполнения, включая динамическое масштабирование потребления ресурсов, управление потоком выполнения, обработку ошибок и искажений вычислений, выполнение локальных оптимизаций. ApplicationMaster выполняет произвольный пользовательский код и может быть написан на любом языке программирования благодаря расширяемым протоколам связи с менеджером ресурсов и менеджером узлов.
* NodeManager (NM) – менеджер узла – агент, запущенный на узле кластера, который отвечает за отслеживание используемых вычислительных ресурсов (CPU, RAM и пр.), управление логами и отправку отчетов об использовании ресурсов планировщику.
* Контейнер (Container) – набор физических ресурсов (ЦП, память, диск, сеть) в одном вычислительном узле кластера.

Принцип работы Hadoop YARN можно описать следующим образом:

* клиентское приложение отправляет запрос в кластер;
* менеджер ресурсов выделяет необходимые ресурсы для контейнера и запускает ApplicationMaster для обслуживания этого приложения;
* ApplicationMaster отправляет запрос менеджеру узла NodeManager, включая контекст запуска контейнера Container Launch Context (CLC);
* ApplicationMaster выделяет контейнеры для приложения в каждом узле и контролирует их работу до завершения работы приложения;
* Для запуска контейнера менеджер узла копирует в локальное хранилище все необходимые зависимости (данные, исполняемые файлы, архивы);
* по завершении задачи мастер приложения отменяет выделенный контейнер в диспетчере ресурсов, завершая жизненный цикл распределенного задания.
* клиент может отслеживать состояние распределенного приложения, обращаясь к менеджеру ресурсов или сразу к мастеру приложения.

**УСТАНОВКА HADOOP**

### Необходимое ПО

Для установки Hadoop необходима операционная система Linux. Далее будет рассмотрена установка и настройка Hadoop для Ubuntu

(возможно использование виртуальной машины).

### Установка Java

Для установки Java на Ubuntu выполните следующую команду:

sudo apt install default-jdk default-jre -y

Для проверки успешной установки узнать версию Java:

java -version

### Создание пользователя для Hadoop и настройка SSH

Создадим нового пользователя. Назовем его *hadoop*:

sudo adduser hadoop

Дадим ему права суперпользователя:

sudo usermod -aG sudo hadoop

Переключимся на созданного пользователя:

sudo su - hadoop

Далее установим OpenSSH сервер и клиент:

sudo apt install openssh-server openssh-client -y

Сгенерируем приватный и публичный ключи:

ssh-keygen -t rsa

На экран будут выведены 2 вопроса:

1. Где сохранить ключ (нажмите enter, чтобы сохранить его в вашем домашнем каталоге)
2. Создайте секретную фразу для ключей (оставьте пустым поле без секретной фразы)

Далее добавим публичный ключ в *authorized\_keys*:

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

Используем команду *chmod* для изменения прав доступа к файлу *authorized\_keys*:

sudo chmod 640 ~/.ssh/authorized\_keys

Проверим настройку SSH:

ssh localhost

Если вы не настроили пароль, все, что вам нужно сделать, это ввести *yes* и нажать *enter,* если вы добавили кодовую фразу для ключей, вам будет предложено ввести ее здесь.

### Установка Apache Hadoop на Ubuntu

Если вы создали пользователя для Hadoop, то войдите под его учетной записью:

sudo su - hadoop

Теперь зайдем на страницу загрузки Apache Hadoop и скопируем ссылку для последней стабильной версии. Используем команду *wget* для установки:

wget https://downloads.apache.org/hadoop/common/sta ble/hadoop-3.3.6.tar.gz

Далее необходимо произвести разархивацию:

tar -xvzf hadoop-3.3.6.tar.gz

Далее перенесем распакованные файлы в */usr/local/hadoop* :

sudo mv hadoop-3.3.6 /usr/local/hadoop

Создадим директорию для хранения логов:

sudo mkdir /usr/local/hadoop/logs

Изменим владельца на *hadoop* для папки */usr/local/hadoop* :

sudo chown -R hadoop:hadoop /usr/local/hadoop

### Настройка Hadoop на Ubuntu

Для начала необходимо открыть *.bashrc* :

sudo nano ~/.bashrc

Переместимся в конец файла комбинацией клавиш *Alt + /* и вставим следующие строки:

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/ native

export PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/ lib/native"

Сохраним изменения и выйдем из текстового редактора *nano*. Для применения изменения выполним команду:

source ~/.bashrc

### Настройка переменных окружения Java

Чтобы работать с Hadoop, вам необходимо включить его основные функции, которые включают YARN, HDFS, MapReduce и параметры проекта, связанные с Hadoop. Для этого определим переменные окружения Java в файле *hadoop-env.sh* :

sudo nano $HADOOP\_HOME/etc/hadoop/hadoop-env.sh

Нажмем комбинацию клавиш *Alt + /* для перемещения в конец файла и укажем путь к Java:

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export HADOOP\_CLASSPATH+=" $HADOOP\_HOME/lib/\*.jar"

Сохраним изменения и выйдем из текстового редактора. Далее сменим рабочую директорию на */usr/local/hadoop/lib* и скачаем javax activation файл:

sudo wget https://jcenter.bintray.com/javax/activati on/javax.activation- api/1.2.0/javax.activation-api-1.2.0.jar

Теперь можно проверить версию Hadoop на Ubuntu:

hadoop version

Далее отредактируем файл *core-site.xml,* чтобы указать URL-адрес для *name node.* Также в нем указывается имя файловой системы (в одном кластере может физически быть несколько файловых систем, однако настроить взаимодействие между ними стандартными средствами не представляется возможным), а также порт, по которому можно к ней обратиться:

Для начала откроем файл *core-site.xml*, используя команду:

sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml

Добавим следующие строки между *<configuration></configuration>*:

<property>

<name>fs.default.name</name>

<value>hdfs://hadoop1:9000</value>

<description>The default file system URI</description>

</property>

Сохраним изменения и выйдем из текстового редактора. Далее создадим директорию для хранения метаданных узлов, используя команду:

sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}

Изменим владельца на *hadoop* для созданной директории:

sudo chown -R hadoop:hadoop /home/hadoop/hdfs

Перейдем к редактированию файла *hdfs-site.xml*. Его настройка необходима для определения местоположения хранения метаданных узлов. Откроем файл в привычном редакторе:

sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

Вставим следующие строки между *<configuration> … </configuration>*:

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>file:///home/hadoop/hdfs/namenode </value>

</property>

<property>

<name>dfs.data.dir</name>

<value>file:///home/hadoop/hdfs/datanode </value>

</property>

Сохраним изменения и выйдем из текстового редактора. Параметр dfs. replication задает количество реплик, которые будут хранится на файловой системе. Также в этом файле прописываются все узлы файловый системы, присутствующие на данной машине. Далее отредактируем файл *mapred-site.xml* для определения значений MapReduce. Откроем файл:

sudo nano $HADOOP\_HOME/etc/hadoop/mapred-site.xml

Вставим следующие строки между *<configuration> … </configuration>*:

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

Сохраним изменения и выйдем из текстового редактора. Далее отредактируем файл *yarn-site.xml* для определения настроек фреймворка управления ресурсами кластера Yarn. Откроем файл:

sudo nano $HADOOP\_HOME/etc/hadoop/yarn-site.xml

Вставим следующие строки между *<configuration> … </configuration>*:

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

Сохраним изменения и выйдем из текстового редактора. Наконец, введем следующую команду для проверки настройки Hadoop и форматирования NameNode HDFS:

hdfs namenode -format

### Запуск Hadoop кластера

Для запуска NameNode и DataNode введем следующую команду:

start-dfs.sh

Далее запустим Node manager и Resource manager:

start-yarn.sh

Для проверки запущенных сервисов, используем команду:

jps

На всех дочерних узлах необходимые демоны запустятся автоматически, используя сконфигурированные подключения по ssh:

Для открытия web-интерфейса введите в адресную строку *http://hadoop1:9870/* в браузере на вашей главной виртуальной машине Ubuntu.

### Установка и настройка кластера Hadoop на Ubuntu

Рассмотрим настройку двух Hadoop кластеров: master и slave. Для начала, если работа велась с использованием VirtualBox, зайдем в настройки машины. Во вкладке «Сеть» необходимо включить сетевой адаптер и в качестве типа подключения выбрать «Виртуальный адаптер хоста». Теперь можно узнать IP, используя команду *ifconfig*.

Нам нужны подчиненные / вторичные узлы. Для этого мы просто клонируем наш мастер / основной, чтобы нам не нужно было все делать заново.

Процесс прост, поэтому просто щелкните правой кнопкой мыши нашу машину и выберите клонировать (рис. 1).

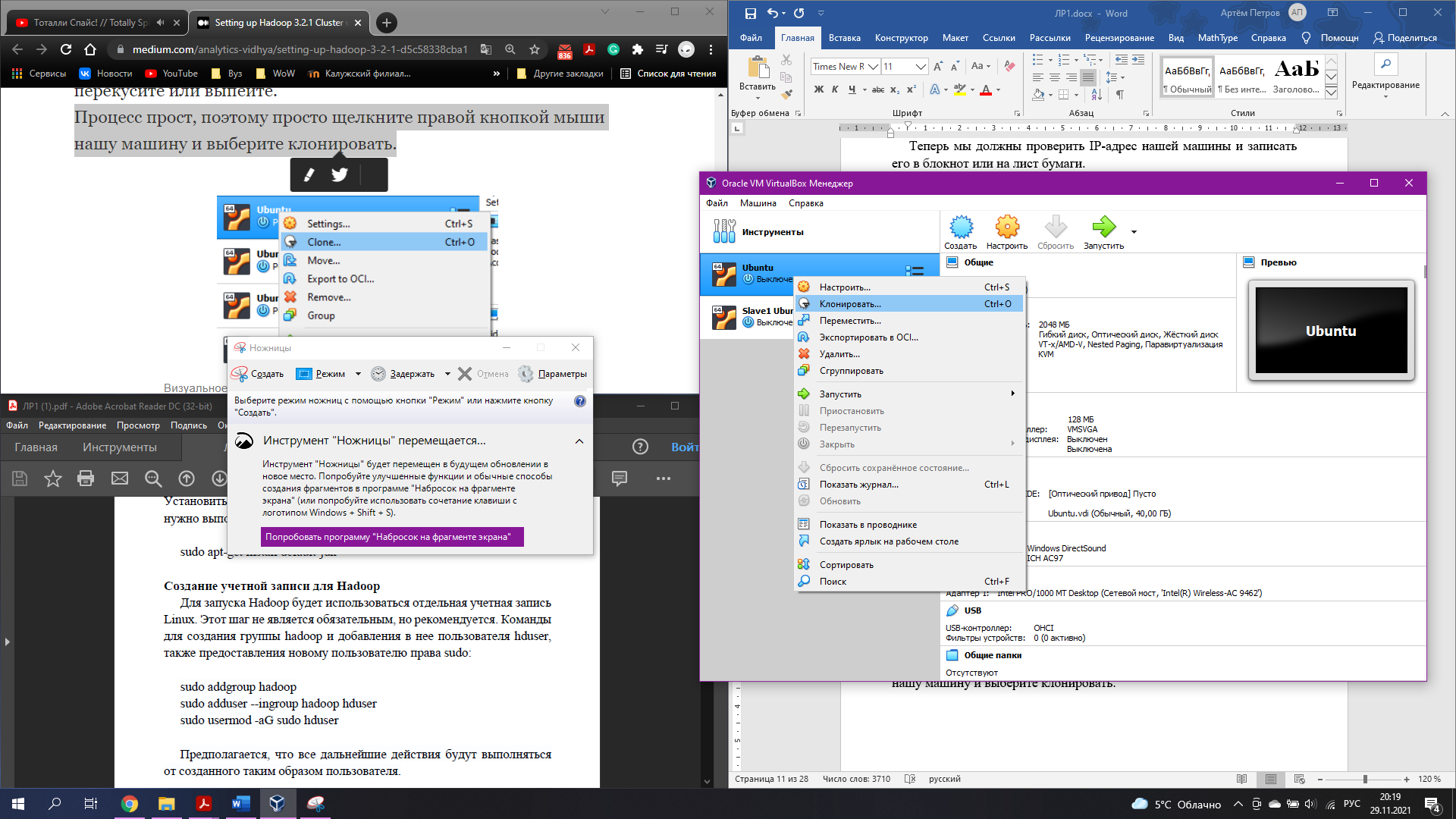


Рис. 1. Клонирование виртуальной машины.

Узлы нашего кластера должны уметь обращаться друг к другу по имени. Для этого на обоих серверах необходимо внести в файл /etc/*hosts* следующие изменения:

#127.0.1.1 hadoop1

192.168.56.101 hadoop1

192.168.56.102 hadoop2

Указанные IP двух машин мы ранее узнали с помощью *ifconfig.*

На мастер-сервере мы создали ключ ssh. Копируем публичный ключ на локальный компьютер:

ssh-copy-id localhost

При первом обращении по SSH будет запрос на принятие сертификата:

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Система запросит ввести пароль для нашего пользователя hadoop. После успешного ввода, мы должны увидеть:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'localhost'"

and check to make sure that only the key(s) you wanted were added.

Теперь скопируем нужные ключи на остальные ноды кластера:

scp -r .ssh hadoop@hadoop2:~

В данном примере мы скопируем каталог .ssh на сервер haddop2, который в нашем примере используются в качестве slave.

Проверим вход в систему по ssh на все серверы — мы должны подключиться без запроса пароля:

ssh localhost

После отключаемся:

exit

Так же подключаемся к другому серверу:

ssh hadoop2

exit

Укажем slave-узлы. Для этого в файле */usr/local/Hadoop/etc/hadoop/workers* оставим лишь строчку:

*hadoop2*

Теперь после запуска Hadoop на двух машинах мы можем убедиться, что slave-узел подключен (рис. 2):

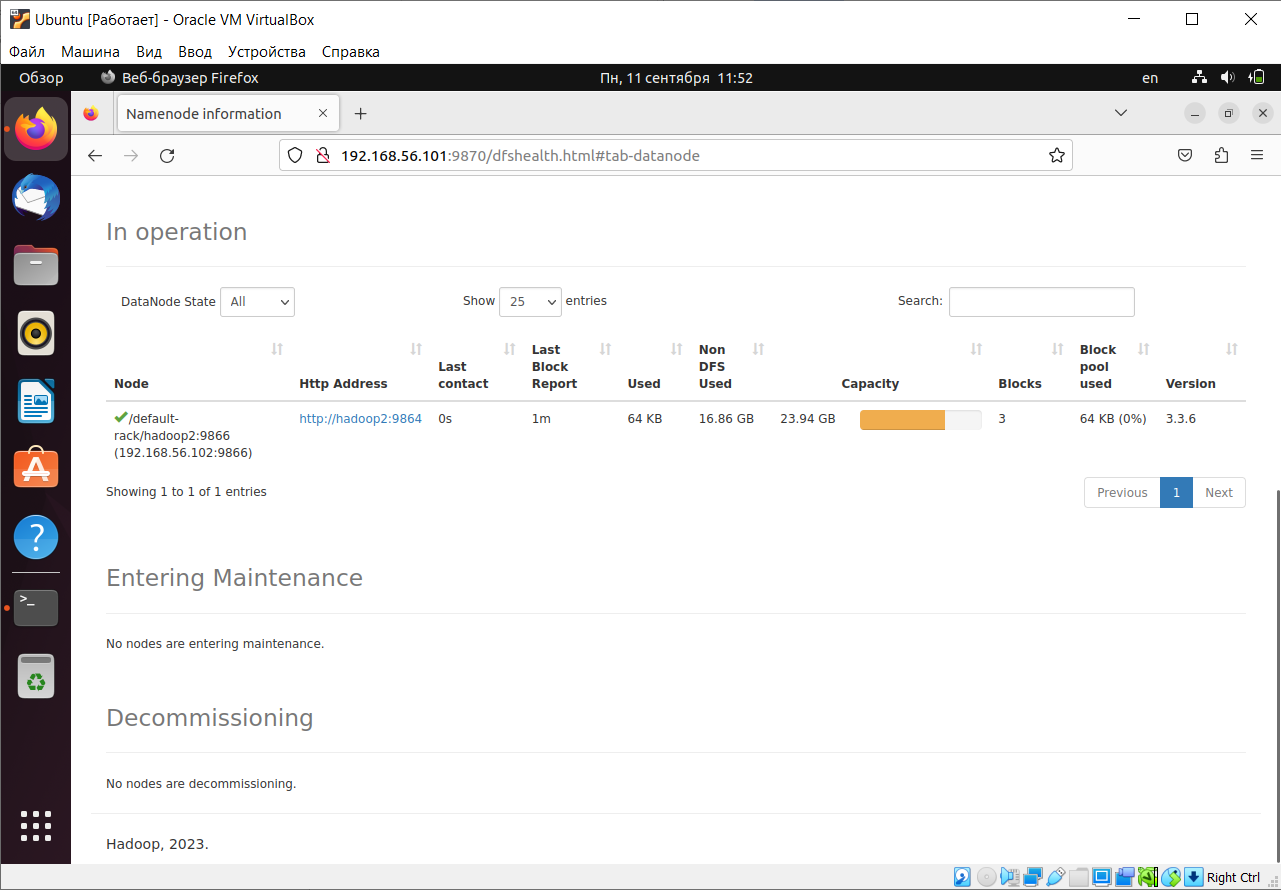


Рис. 2. Подключенный slave-узел.

# РАБОТА С HDFS

Для работы с hdfs в терминале существует набор команд, схожих с командами работы с файловой системой в unix-системах (просмотр файлов, создание, удаление, просмотр и т.д.), для их вызова необходимо указать исполняемый файл hadoop и в виде ключа передать команду, например, для создания каталога необходимо выполнить команду:

hadoop fs -mkdir /usr/hduser

Полный перечень доступных команд можно просмотреть в прилагаемой к дистрибутиву hadoop документации.

Следует выделить следующие команды, не имеющие аналогов среди стандартных команд работы с файловой системой:

-copyFromLocal <localsrc> ... <dst>] – позволяет скопировать файл из локальной файловой системы в hdfs

-copyToLocal [<src> ... <localdst>] – обратная команда, позволяющая скопировать файл из hdfs в локальную файловую систему.

Также, как упоминалось выше, возможно управление файлами через web-интерфейс.

Приведем основные команды.

Просмотр содержимого директории и установленных прав доступа:

hdfs dfs -ls /

Чтобы просмотреть абсолютно все находящиеся внутри файлы, можно добавить в команду ключ *-R*.

Создание директории:

hdfs dfs -mkdir /user

Создание файла:

hdfs dfs -touchz /file.txt

Заполнение файла:

hadoop fs -appendToFile - /file.txt

Для прекращения записи необходимо применить комбинацию клавиш *Ctrl + C*.

Чтение файла:

hdfs dfs -cat /file.txt

Копирование файла на локальную систему:

hdfs dfs -get /file.txt ./

Отправка файла на HDFS:

hdfs dfs -put new\_file.txt /

Перемещение файла в пределах HDFS:

hdfs dfs -mv /file.txt /folder/file.txt

Копирование файла в пределах HDFS:

hdfs dfs -cp /file.txt /copy.txt

Удаление файла:

hdfs dfs -rm /file.txt

Для удаления директории также необходимо добавить ключ *-R*.

Изменение прав доступа:

hadoop fs -chmod 770 /file.txt

Проверка наличия файла:

hadoop fs -test -e /file.txt

Чтение времени последнего изменения:

hdfs dfs -stat /file.txt

### HDFS API

Помимо управления файлами через консольный режим доступен Java hdfs api, содержащий набор функций для управления файлами в java программе. Все необходимые библиотеки входят в дитрибутив hadoop, но также могут быть скачены, используя сборщик проектов maven, путем указания следующих зависимостей в pom-файле:

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-hdfs</artifactId>

<version>2.9.0</version>

</dependency>

После подключения всех необходимых библиотек необходимо сконфигурировать файловую систему:

Configuration conf = new Configuration();

conf.set("fs.default.name", hdfsPath);

После этого с hdfs можно работать как и с обычной файловой системой, например, для создания каталога необходимо использовать команду:

client.mkdir(name, conf);

Помимо java hdfs api, распространяемого вместе с дистрибутивом hadoop, существует множество библиотек для работы с hdfs, используя различные языки программирования, такие как python, scala и др.

**ПРИМЕР ЗАДАЧ**

**Пример 1. Программа вывода списка файлов для заданного каталога.**

Решение задачи на языке программирования Python:

import subprocess

path = "/data"

args = "hdfs dfs -ls "+path+" | awk '{print $8}'"

proc = subprocess.Popen(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)

s\_output, s\_err = proc.communicate()

all\_dart\_dirs = s\_output.split()

Результат выполнения программы (рис. 3):

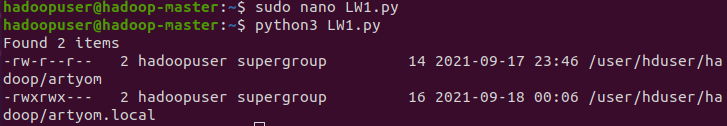


Рис. 3. Результат выполнения программы.

**Пример 2. Программа копирования файла в систему HDFS.**

Решение задачи на языке программирования Python:

import argparse

from hdfs import insecureclient

parser = argparse.argumentparser()

parser.add\_argument(“--local”)

parser.add\_argument(“--dst”)

args=parser.parse\_args()

local=args.local

dst=args.dst

web\_hdfs\_interface=insecureclient(“http://192.168.1.70:9870”, user=”hadoopuser”)

web\_hdfs\_interface.upload(dst, local)

Результат выполнения программы (рис. 4):

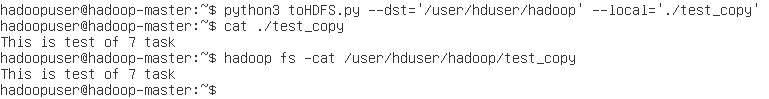


Рис. 4. Результат выполнения программы.

# ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Для всех вариантов настроить кластер Hadoop, состоящий из двух серверов, изучить команды HDFS для работы с файлами и выполнить следующие задания:

1. Проверить существует ли директория /user/hduser в HDFS, если нет, то создать. Создать директорию /user/hduser/Hadoop
2. Создать файл в директории /user/hduser/hadoop, название файла

– ваше имя и группа. После создания файла, все, что вы вводите в консоль должно сохраниться в файле. Ввести несколько строк и сохранить.

1. Убедиться в существовании файла через web-интерфейс.
2. Перенести файл в локальную файловую систему.
3. Создать новый текстовый файл в локальной файловой системе. Перенести файл в HDFS. Убедиться в существовании файла через web-интерфейс.
4. Просмотреть права доступа на файл. Изменить права доступа к файлу, чтобы только владелец и члены группы имели полный контроль над файлом.
5. Написать программу на каком-либо языке высокого уровня для решения задачи, указанной в варианте.

# ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Программа может быть реализована на любом языке высокого уровня, для которого существует поддержка работы с HDFS (Java, Python, Scala или др.). Имена файлов должны передаваться приложению в качестве ключей при вызове в терминале.

# ВАРИАНТЫ ЗАДАНИЙ

1. Напишите программу, которая будет выводить на экран список подкаталогов и файлов для заданного каталога в HDFS.
2. Напишите программу, которая на вход получает путь в HDFS, а на выходе выводит количество каталогов по заданному пути.
3. Напишите программу, которая будет выводить на экран содержимое файла в HDFS.
4. Напишите программу, которая на вход получает путь в HDFS и строку с именем каталога, если такой каталог есть, то программа его удалит. В случае, если каталог не существует - выдать сообщение.
5. Напишите программу, которая будет копировать файл из локальной файловой системы в систему HDFS.
6. Напишите программу, которая на вход получает путь в HDFS и строку с именем каталога, если такого каталога нет, то программа его создаст. В случае, если каталог существует - выдать сообщение.
7. Напишите программу, которая будет рекурсивно выводить на экран список подкаталогов и файлов для заданного каталога в HDFS.
8. Напишите программу, которая на вход получает путь в HDFS, строку с именем файла и строку с предложением. Программа выводит в консоль содержимое файла до добавления предложения и после.
9. Напишите программу, которая будет выводить на экран список всех подкаталогов и файлов в заданной директории HDFS, которые были изменены в промежуток между start\_ts и end\_ts (передаются через параметры командной строки).
10. Напишите программу, которая переместит файл из одного каталога в другой в HDFS.
11. Напишите программу, которая будет копировать все содержимое каталога из файловой системы HDFS в локальную.
12. Напишите программу, которая на вход получает путь и число и изменяет коэффициент репликации всех файлов в каталоге по заданному пути на введенное число.
13. Напишите программу, которая будет принимать 2 входных аргумента – путь в локальной файловой системе и путь в HDFS. Программа должна проверить существование файлов в обоих файловых системах. Если в одной из них файл не существует, то программа должна скопировать его из второй файловой системы. Если файлы существуют в обоих файловых системах, то сохранить в обоих системах только файл, который был изменен позже.
14. Напишите программу, которая получает на вход путь в HDFS, находит в каталоге файл, который был создан позже других, и удаляет его.
15. Напишите программу, которая будет для заданного каталога в HDFS добавлять в содержимое всех файлов из этого каталога текущую дату.
16. Напишите программу, которая на вход получает путь в HDFS и число и создает заданное количество файлов в каталоге.
17. Напишите программу, которая будет сравнивать содержимое двух текстовых файлов в HDFS.
18. Напишите программу, которая будет сравнивать количество файлов/подкаталогов в двух каталогах в HDFS и удалит каталог с наименьшим количеством файлов/подкаталогов.
19. Напишите программу, которая рекурсивно удаляет все файлы из заданного каталога HDFS.
20. Напишите программу, которая скопирует все содержимое каталога из файловой системы HDFS в локальную и удалит каталог из HDFS.

**КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ**

1. Раскройте назначение платформы Hadoop.
2. Перечислите основные модули Hadoop и опишите их назначение.
3. Раскройте область применения платформы Hadoop.
4. Раскройте назначение HDFS.
5. Опишите архитектуру HDFS.
6. Раскройте назначение YARN.
7. Опишите архитектуру YARN.
8. Изложите принципы работы Hadoop YARN.
9. Перечислите ПО необходимое для установки Hadoop.
10. Перечислите основные этапы установки Hadoop.
11. Перечислите некоторые команды для работы с HDFS и опишите их назначение.
12. Опишите назначение HDFS API.

# ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 3 занятия (6 академических часов: 5 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), этапы выполнения работы (со скриншотами), результаты выполнения работы. выводы.

# ОСНОВНАЯ ЛИТЕРАТУРА

1. Федин Ф.О. Анализ данных. Часть 1. Подготовка данных к анализу [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 204 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/26444.html
2. Федин Ф.О. Анализ данных. Часть 2. Инструменты Data Mining [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 308 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/26445.html
3. Чубукова, И.А. Data Mining [Электронный ресурс] : учеб. пособие

— Электрон. дан. — Москва : , 2016. — 470 с. — Режим доступа: https://e.lanbook.com/book/100582. — Загл. с экрана.

1. Воронова Л.И. Big Data. Методы и средства анализа [Электронный ресурс] : учебное пособие / Л.И. Воронова, В.И. Воронов. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 33 c. — 2227-8397. — Режим доступа:<http://www.iprbookshop.ru/61463.html>5. Юре, Л. Анализ больших наборов данных [Электронный ресурс] / Л.

Юре, Р. Ананд, Д.У. Джеффри. — Электрон. дан. — Москва : ДМК

Пресс, 2016. — 498 с. — Режим доступа: https://e.lanbook.com/book/93571. — Загл. с экрана.

# ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

1. Волкова Т.В. Разработка систем распределенной обработки данных [Электронный ресурс] : учебно-методическое пособие / Т.В. Волкова, Л.Ф. Насейкина. — Электрон. текстовые данные. —

Оренбург: Оренбургский государственный университет, ЭБС АСВ,

2012. — 330 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/30127.html

1. Кухаренко Б.Г. Интеллектуальные системы и технологии

[Электронный ресурс] : учебное пособие / Б.Г. Кухаренко. — Электрон. текстовые данные. — М. : Московская государственная академия водного транспорта, 2015. — 116 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/47933.html

1. Воронова Л.И. Интеллектуальные базы данных [Электронный ресурс] : учебное пособие / Л.И. Воронова. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2013. — 35 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/63324.html
2. Николаев Е.И. Базы данных в высокопроизводительных информационных системах [Электронный ресурс] : учебное пособие / Е.И. Николаев. — Электрон. текстовые данные. —

Ставрополь: Северо-Кавказский федеральный университет, 2016.

— 163 c. — 2227-8397. — Режим доступа:

http://www.iprbookshop.ru/69375.html

**Электронные ресурсы:**

1. http://hadoop.apache.org/ (англ.)
2. <https://learnubuntu.com/install-hadoop/>
3. https://www.dmosk.ru/miniinstruktions.php?mini=hadoop-install