Что такое Hadoop?

**Hadoop**– это проект с открытым исходным кодом, находящийся под управлением Apache Software Foundation (<http://hadoop.apache.org/>). 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 (англ. HDFS), которая отвечает за хранение данных на кластере Hadoop;

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

На основе этих ключевых компонентов создано несколько подпроектов, таких как Pig, Hive, HBase и т.д.

HDFS

**HDFS**, или Распределенная файловая система Hadoop – это основная система хранения данных, используемая приложениями Hadoop. HDFS многократно копирует блоки данных и распределяет эти копии по вычислительным узлам кластера, тем самым обеспечивая высокую надежность и скорость вычислений:

- Данные распределяются по нескольким машинам во время загрузки.

- HDFS оптимизирована больше для выполнения потоковых считываний файлов, нежели для нерегулярных, произвольных считываний.

- Файлы в системе HDFS пишутся однократно и внесение никаких произвольных записей в файлы не допускается.

- Приложения могут считывать и писать файлы Распределенной файловой системы напрямую через программный интерфейс Java.

MapReduce

**MapReduce**– это модель программирования и каркас для написания приложений, предназначенных для высокоскоростной обработки больших объемов данных на больших параллельных кластерах вычислительных узлов:

- обеспечивает автоматическое распараллеливание и распределение задач;

- имеет встроенные механизмы сохранения устойчивости и работоспособности при сбое отдельных эелементов;

- обеспечивает чистый уровень абстракции для программистов.

Hadoop и корпоративные системы

Системы управления реляционными базами данных (СУРБД, или англ. RDBMS) используются в системах больших компаний по стандартной схеме:

- Интерактивная система СУРБД обрабатывает запросы, поступающие с веб-сайта или других пользовательских приложений.

- Затем данные извлекаются из реляционной базы и загружаются в хранилище данных для дальнейшей обработки и архивации.

- Данные обычно денормализуются в OLAP-куб.

К сожалению, современные системы СУРБД не могут вместить все то огромное количество данных, которое создается в больших компаниях, и тогда возникает необходимость идти на компромисс: данные либо только частично копируются в СУРБД, либо удаляются через определенное время. Необходимость таких компромиссных решений отпадает, если в качестве промежуточного слоя между интерактивной базой данных и хранилищем данных используется Hadoop:

- Производительность обработки данных возрастает пропорционально увеличению объема хранилища данных, в то время как в высокопроизводительных серверах рост количества данных и изменение производительности непропорциональны.

- При использовании Hadoop, чтобы повысить производительность обработки, достаточно просто добавить новые узлы в хранилище данных.

- Hadoop может хранить и обрабатывать многие петабайты данных.

Однако Hadoop имеет ряд серьезных ограничений, и потому Hadoop не может использоваться в качестве операционной базы данных:

- Для выполнения самой быстрой задачи в Hadoop все же требуется несколько секунд.

- Не допускается внесение никаких изменений в данные, которые хранятся в системе HDFS.

- Hadoop не поддерживает транзакции.

Hadoop и СУРБД

Системы управления реляционными базами данных (СУРБД) имеют много достоинств:

- они способны оперировать сложными транзакциями;

- в них могут обрабатываться сотни тысяч запросов в секунду;

- результаты выдаются в реальном времени;

- применяется простой, но эффективный язык запросов.

Однако у СУРБД есть и слабые стороны:

- Схема данных определяется до того, как данные импортируются;

- Предельная вместимость СУРБД достигает сотен терабайтов;

- Предельное количество данных в одном запросе составляет десятки терабайтов.

Hadoop и файловые хранилища

Бизнес-данные больших компаний часто хранятся на больших файловых серверах типа NetApp, EMC и т.п., которые обеспечивают быстрый и произвольный доступ к данным и могут поддерживать одновременно большое количество клиентских приложений. Однако когда речь идет о хранении петабайтов данных, цена за хранение одного терабайта данных может сильно возрастать. В этом случае Hadoop представляет собой действительно удачную альтернативу файловым хранилищам, с условием, что произвольный доступ к данным возможно заменить на последовательные считывания, а изменения данных допустимо ограничить только присоединенными записями.

Hive и Pig

<http://hive.apache.org/>

<http://pig.apache.org/>

MapReduce является очень мощным инструментом обработки данных, однако он может оказаться довольно сложным для создания и поддержания, в то время как во многих компаниях работают бизнес-аналитики, которые хорошо умеют писать запросы на языке SQL, но не умеют писать код на Java. Также, во многих организациях есть программисты, умеющие писать код на скриптовых языках. Hive и Pig – это два проекта, которые разрабатывались независимо друг от друга и которые предназначены для того, чтобы помочь таким аналитикам и программистам эффективно использовать MapReduce для анализа больших массивов данных.

**Hive**– это надстройка над Hadoop для того, чтобы облегчить выполнение таких задач, как суммирование данных, непрограммируемые запросы и анализ больших наборов данных:

- Hive может быть использован теми, кто знает язык SQL.

- Hive создает задания MapReduce, которые исполняются на кластере Hadoop.

- Определения таблиц в Hive надстраиваются над данными в HDFS.

**Pig**– это платформа, предназначенная для анализа больших наборов данных и состоящая из языка высокого уровня для написания программ анализа данных и инфраструктуры для запуска этих программ. Язык характеризуется  относительно простым синтаксисом. Написанные  сценарии скрыто преобразуются в задачи MapReduce, которые исполняются на кластере Hadoop.

HBase

<http://hbase.apache.org/>

**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 доступа к данным в HBase имеет следующие ограничения:

- поиск ряда по одному ключу;

- не поддерживаются транзакции;

- доступны только операции по одной строке.

# Инструкция по настройке настройка

#### Предварительные настройки

В качестве операционной системы для нашего кластера выбираем **Debian 8.0**, но при минимальных изменениях можно будет проделать все шаги и на другой ОС.  
  
Все узлы будут работать на VirtualBox. Системные настройки для виртуальной машины я выставлял небольшие. Всего 8 GB пространства для жёсткого диска, одно ядро и 512 Мб памяти. Виртуальная машина также оснащена двумя сетевыми адаптерами: один NAT, а другой для внутренней сети.  
  
После того, как была скачена и установлена операционная система, необходимо обновиться и установить ssh и rsync:

sudo apt-get update && sudo apt-get upgrade

sudo apt-get install ssh

sudo apt-get install rsync

#### Java

Для работы Hadoop можно использовать либо 6 или 7 версию.  
В данной статье будем работать с OpenJDK 7 версии:

$ sudo apt-get install openjdk-7-jdk  
  
Хотя можно использовать версию от Oracle.

**И как?**

Очищаем ОС от всех зависимостей OpenJDK sudo apt-get purge openjdk\*  
Устанавливаем python-software-properties который позволит добавлять новые PPA:

sudo apt-get install python-software-properties

Добавляем PPA с [launchpad.net/~webupd8team/+archive/java](https://launchpad.net/~webupd8team/+archive/java)

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java7-installer

#### Создание отдельной учетной записи для запуска Hadoop

Мы будем использовать выделенную учетную запись для запуска Hadoop. Это не обязательно, но рекомендуется. Также предоставим новому пользователю права sudo, чтобы облегчить себе жизнь в будущем.

sudo addgroup hadoop

sudo adduser --ingroup hadoop hduser

sudo usermod -aG sudo hduser

Во время создания нового пользователя, необходимо будет ввести ему пароль.

#### /etc/hosts

Нам необходимо, чтобы все узлы могли легко обращаться друг к другу. В большом кластере желательно использовать dns сервер, но для нашей маленькой конфигурации подойдет файл hosts. В нем мы будем описывать соответствие ip-адреса узла к его имени в сети. Для одного узла ваш файл должен выглядеть примерно так:

127.0.0.1 localhost

*# The following lines are desirable for IPv6 capable hosts*

::1 ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

192.168.0.1 master

#### SSH

Для управления узлами кластера hadoop необходим доступ по ssh. Для созданного пользователя hduser предоставить доступ к master.  
Для начала необходимо сгенерировать новый ssh ключ:

ssh-keygen -t rsa -P ""

Во время создания ключа будет запрошен пароль. Сейчас можно его не вводить.  
  
Следующим шагом необходимо добавить созданный ключ в список авторизованных:

cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys

Проверяем работоспособность, подключившись к себе:

ssh master

#### Отключение IPv6

Если не отключить IPv6, то в последствии можно получить много проблем.  
Для отключения IPv6 в Ubuntu 12.04 / 12.10 / 13.04 нужно отредактировать файл sysctl.conf:

sudo vim /etc/sysctl.conf

Добавляем следующие параметры:

*# IPv6*

net.ipv6.conf.all.disable\_ipv6 = 1

net.ipv6.conf.default.disable\_ipv6 = 1

net.ipv6.conf.lo.disable\_ipv6 = 1

Сохраняем и перезагружаем операционную систему.

#### Установка Apache Hadoop

Скачаем необходимые файлы.  
Актуальные версии фреймворка располагаются по адресу: [www.apache.org/dyn/closer.cgi/hadoop/common](http://www.apache.org/dyn/closer.cgi/hadoop/common/)

Создадим папку downloads в корневом каталоге и скачаем последнюю версию:

sudo mkdir /downloads

cd downloads/

sudo wget http://apache-mirror.rbc.ru/pub/apache/hadoop/common/stable/hadoop-2.7.3.tar.gz  
  
Распакуем содержимое пакета в /usr/local/, переименуем папку и выдадим пользователю hduser права создателя:

sudo mv /downloads/hadoop-2.7.3.tar.gz /usr/local/

cd /usr/local/

sudo tar xzf hadoop-2.7.3.tar.gz

sudo mv hadoop-2.7.3 hadoop

chown -R hduser:hadoop hadoop

#### Обновление $HOME/.bashrc

Для удобства, добавим в .bashrc список переменных:

*#Hadoop variables*

export JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

export HADOOP\_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/bin

export PATH=$PATH:$HADOOP\_INSTALL/sbin

export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL

export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL

export YARN\_HOME=$HADOOP\_INSTALL

На этом шаге заканчиваются предварительные подготовки.

#### Настройка Apache Hadoop

Все последующая работа будет вестись из папки /usr/local/hadoop.  
Откроем etc/hadoop/hadoop-env.sh и зададим JAVA\_HOME.

vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Опишем, какие у нас будут узлы в кластере в файле etc/hadoop/slaves

master  
  
Этот файл может располагаться только на главном узле. Все новые узлы необходимо описывать здесь.

Основные настройки системы располагаются в etc/hadoop/core-site.xml:

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

</configuration>

Настройки HDFS лежат в etc/hadoop/hdfs-site.xml:

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

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

<value>file:/usr/local/hadoop/tmp/hdfs/namenode</value>

</property>

<property>

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

<value>file:/usr/local/hadoop/tmp/hdfs/datanode</value>

</property>

</configuration>

Здесь параметр dfs.replication задает количество реплик, которые будут хранится на файловой системе. По умолчанию его значение равно   
  
3. Оно не может быть больше, чем количество узлов в кластере.  
Параметры dfs.namenode.name.dir и dfs.datanode.data.dir задают пути, где будут физически располагаться данные и информация в HDFS. Необходимо заранее создать папку tmp.  
  
Сообщим нашему кластеру, что мы желаем использовать YARN. Для этого изменим etc/hadoop/mapred-site.xml:

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

Все настройки по работе YARN описываются в файле etc/hadoop/yarn-site.xml:

<configuration>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8032</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>master:8033</value>

</property>

</configuration>

Настройки resourcemanager нужны для того, чтобы все узлы кластера можно было видеть в панели управления.  
  
Отформатируем HDFS:

bin/hdfs namenode –format

Запустим hadoop службы:

sbin/start-dfs.sh

sbin/start-yarn.sh

Необходимо убедиться, что запущены следующие java-процессы:

hduser@master:/usr/local/hadoop$ jps

4868 SecondaryNameNode

5243 NodeManager

5035 ResourceManager

4409 NameNode

4622 DataNode

5517 Jps

Протестировать работу кластера можно при помощи стандартных примеров:

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar

Теперь у нас есть готовый образ, который послужит основой для создания кластера.  
  
Далее можно создать требуемое количество копий нашего образа.  
  
На копиях необходимо настроить сеть. Необходимо сгенерировать новые MAC-адреса для сетевых интерфейсов и выдать и на них необходимые ip-адреса.

Поправить файл /etc/hosts на всех узлах кластера так, чтобы в нем были прописаны все соответствия.

**Пример**

127.0.0.1 localhost

*# The following lines are desirable for IPv6 capable hosts*

::1 ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

192.168.0.1 master

192.168.0.2 slave1

192.168.0.3 slave2

Для удобства, изменить имена новых узлов на slave1 и slave2.

**Как?**

Необходимо изменить два файла: /etc/hostname и /etc/hosts.

Сгенерируйте на узлах новые SSH-ключи и добавьте их все в список авторизованных на узле master.  
  
На каждом узле кластера изменим значения параметра dfs.replication в etc/hadoop/hdfs-site.xml. Например, выставим везде значение 3.

**etc/hadoop/hdfs-site.xml**

<configuration>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

</configuration>

Добавим на узле master новые узлы в файл etc/hadoop/slaves:

master

slave1

slave2

Когда все настройки прописаны, то на главном узле можно запустить наш кластер.

bin/hdfs namenode –format

sbin/start-dfs.sh

sbin/start-yarn.sh

На slave-узлах должны запуститься следующие процессы:

hduser@slave1:/usr/local/hadoop$ jps

1748 Jps

1664 NodeManager

1448 DataNode

Теперь у нас есть свой мини-кластер.  
  
Давайте запустим задачу Word Count.  
Для этого нам потребуется загрузить в HDFS несколько текстовых файлов.  
Для примера, я взял книги в формате txt с сайта [Free ebooks — Project Gutenberg](http://www.gutenberg.org/).

**Тестовые файлы**

cd /home/hduser

mkdir books

cd books

wget http://www.gutenberg.org/cache/epub/20417/pg20417.txt

wget http://www.gutenberg.org/cache/epub/5000/pg5000.txt

wget http://www.gutenberg.org/cache/epub/4300/pg4300.txt

wget http://www.gutenberg.org/cache/epub/972/pg972.txt

wget http://www.gutenberg.org/cache/epub/132/pg132.txt

wget http://www.gutenberg.org/cache/epub/1661/pg1661.txt

wget http://www.gutenberg.org/cache/epub/19699/pg19699.txt

Перенесем наши файлы в HDFS:

cd /usr/local/hadoop

bin/hdfs dfs -mkdir /**in**

bin/hdfs dfs -copyFromLocal /home/hduser/books/\* /**in**

bin/hdfs dfs -ls /**in**  
  
Запустим Word Count:

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount /**in** /out  
  
Отслеживать работу можно через консоль, а можно через веб-интерфейс ResourceManager'а по адресу [master](http://master/):8088/cluster/apps/  
  
По завершению работы, результат будет располагаться в папке /out в HDFS.   
Для того, чтобы скачать его на локальную файловую систему выполним:

bin/hdfs dfs -copyToLocal /out /home/hduser/