электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Установка и настройка кластера Hadoop на базе контейнерной виртуализации LXC

 # 04, апрель 2015

к.т.н., доцент Виноградова М. В.[1,*], Анисимов Н. А.[1]

УДК: 004.75

[1]Россия, МГТУ им. Н.Э. Баумана
[*]vinogradova.m@bmstu.ru

Введение

В эпоху общедоступности сети Интернет, которая приводит к большим нагрузкам на веб-серверы и серверы приложений, растет необходимость в гибком управлении мощностями и ресурсами работающей системы с целью предоставления качественного сервиса конечному клиенту, а также с целью минимизации стоимости обслуживания [1]. И наоборот, при не очень высоких нагрузках, имея в распоряжении всего один физических сервер, требуется удобно и безопасно изолировать друг от друга различные процессы внутри одного физического узла [2]. В обоих случаях разработчикам программ необходима среда для создания или отладки распределенных приложений.

Разработка распределенных приложений актуальна для многих сфер деятельности, например, при построении архива документов в организации [3], или при создании системы распознавания лиц пассажиров в режиме реального времени [4], или при сборе и обработке больших объемов данных в целях анализа дорожной ситуации [5]. Поскольку речь идет о больших и дорогостоящих системах, то важную роль играет качество проектирования и корректность выбранных технологий. Поэтому на ранних стадиях разработки необходимо иметь возможность оценить производительность работы компонентов или эффективность технических решений.

Для выбора проектных решений можно использовать экспертные оценки [6] или применять средства мониторинга работы устройств [7]. Указанные подходы позволяют выбирать программное и аппаратное обеспечение, а также конфигурацию сети [8]. Для оценки эффективности проектных и архитектурных решений удобно создавать тестовые стенды, на которых развертывать прототипы системы. Такие прототипы будут являться вычислительными кластерами.

Для поддержки работы вычислительных кластеров, выполняющих распределенную обработку на множестве узлов, разработаны специальные средства и технологии. Одной

из наиболее эффективных и широко используемых является Hadoop [9]. Hadoop – это набор технологий, утилит, фреймворков и библиотек для разработки и и выполнения распределенных задач, работающих на множестве узлов, объединенных в кластер. Поскольку Hadoop часто используется совместно с большим количеством не очень мощных узлов, то возможно развернуть его на одном узле [10]. Это позволит разработчикам проверить работоспособность создаваемых программ или оценить эффективность их работы при небольших затратах на аппаратное обеспечение.

Однако, Hadoop в пределах одного физического узла позволяет запустить либо один логический узел, либо псевдо-кластер [11]. Это ограничение не позволяет полноценно протестировать архитектуру и запуск вычислительных задач.

Для запуска полнофункционального вычислительного кластера на одном физическом узле хорошо подходит виртуализация [12], которая позволяет горизонтально масштабировать сервисы в пределах одного физического узла. Виртуализация дает возможность запускать на одном компьютере несколько экземпляров операционных систем, каждый из которых работает независимо один от другого, и объединять их в виртуальную сеть.

Поскольку создание и настройка вычислительного кластера с Hadoop на одном компьютере является нетривиальной задачей, то была выявлена необходимость разработки и практической апробации технологии по ее выполнению.

В данной работе приведено описание технологии развертывания на одном физическом узле полноценного кластера Hadoop, состоящего из трех логических узлов – контейнеров. При помощи контейнерной виртуализации [13] возможно реализовать три абсолютно изолированных логических узла, которые затем настраиваются и работают как полноценный кластер. Технология рассматривается для ОС из семейства Linux. Данная ОС (Linux Ubuntu) была выбрана как наиболее популярная среди разработчиков распределенных приложений.

В результате использования предлагаемой технологии будет получена инфраструктура, в которой можно разрабатывать, исследовать или тестировать распределенные программные приложения. Построенную инфраструктуру можно использовать

Контейнерная виртуализация

Прежде всего рассмотрим принципы и технологии виртуализации.

В последнее время наибольшей популярностью обладает контейнерная виртуализация [14] - виртуализация на уровне операционной системы (ОС). Она позволяет использовать ресурсы физического сервера более рационально за счет отсутствия виртуализации ядер запускаемых контейнеров. Рассмотрим особенности ее реализации.

На физическом сервере работает главная ОС, ядро которой поддерживает несколько других изолированных пространств процессов, которые и называют контейнерами. При этом, аналогично более низкоуровневым методам виртуализации, можно выделять отдельным пространствам ограниченные ресурсы, например, выделяемую оперативную па-

мять (ОП), задавать квоту на доступ к центральному процессору (ЦП) и так далее. Конечно, есть небольшое ограничение – виртуализируемое пространство (содержимое контейнера) должно быть совместимо с ядром, на котором оно виртуализируется.

Контейнерная виртуализация LXC (англ. Linux Containers) [15] запускает изолированные экземпляры операционной системы Linux на одном физическом узле с ОС Linux. Изолированные контейнеры не имеют собственного ядра, вместо этого задачи выполняются в ядре главной ОС физического узла. Каждый созданный контейнер имеет собственное пространство процессов и сетевой стек. При необходимости каждому контейнеру можно задать ограничения на использование ресурсов физического узла, также можно задать приоритеты. Каждый контейнер во включенном состоянии имеет работающую службу SSH, позволяющую к нему подключаться и выполнять любые действия, как в обычной ОС.

Контейнерная виртуализация весьма эффективна для развертывания Hadoop на одном узле. Рассмотрим технологию и состав Hadoop.

Технология Hadoop

Hadoop – это стек технологий, набор утилит, фреймворков и библиотек для разработки и выполнения распределенных задач, работающий на большом количестве узлов, объединенных в кластер. Используется преимущественно с очень большим количеством узлов (тысячи узлов), не очень мощных. Включает в себя несколько основных модулей:

- Hadoop Common – набор общих утилит и программное обеспечение (ПО) для работы с инфраструктурой и всем стеком;
- HDFS – распределённая файловая система;
- YARN – система планирования заданий и управления кластером;
- MapReduce – платформа для создания и выполнения распределенных вычислений.

Ключевой компонент – это HDFS, который представляет файловую систему, распределенную по всему кластеру. При сохранении файла в HDFS, он автоматически разбивается на блоки заданного размера, и каждый блок с заданным коэффициентом репликации сохраняется на разные узлы кластера. Коэффициент репликации задает количество копий файла в кластере. Например, если он равен трем, то блок файла будет сохранен на трех разных узлах, выбранных с учетом уже занятого места в системе хранения (жестком диске). То есть, кластер автоматически балансирует занятое место, распределяя блоки равномерно. Репликация блоков позволяет сделать систему хранения более отказоустойчивой, например, при отключении очередного узла, если блок останется доступным на другом узле, кластер продолжит свою работу.

Следующий компонент — это платформа MapReduce, которая позволяет запускать распределенные вычислительные задачи, выполняющиеся в два этапа. Этап Map выполняется на каждом узле, в котором есть необходимый для выполнения задачи блок данных, и предназначен для фильтрации, сортировки, группировки данных. Затем выполняется этап Reduce, обрабатывающий сгруппированные ранее на этапе Map данные.

Типичная архитектура кластера Hadoop представлена на рис. 1.

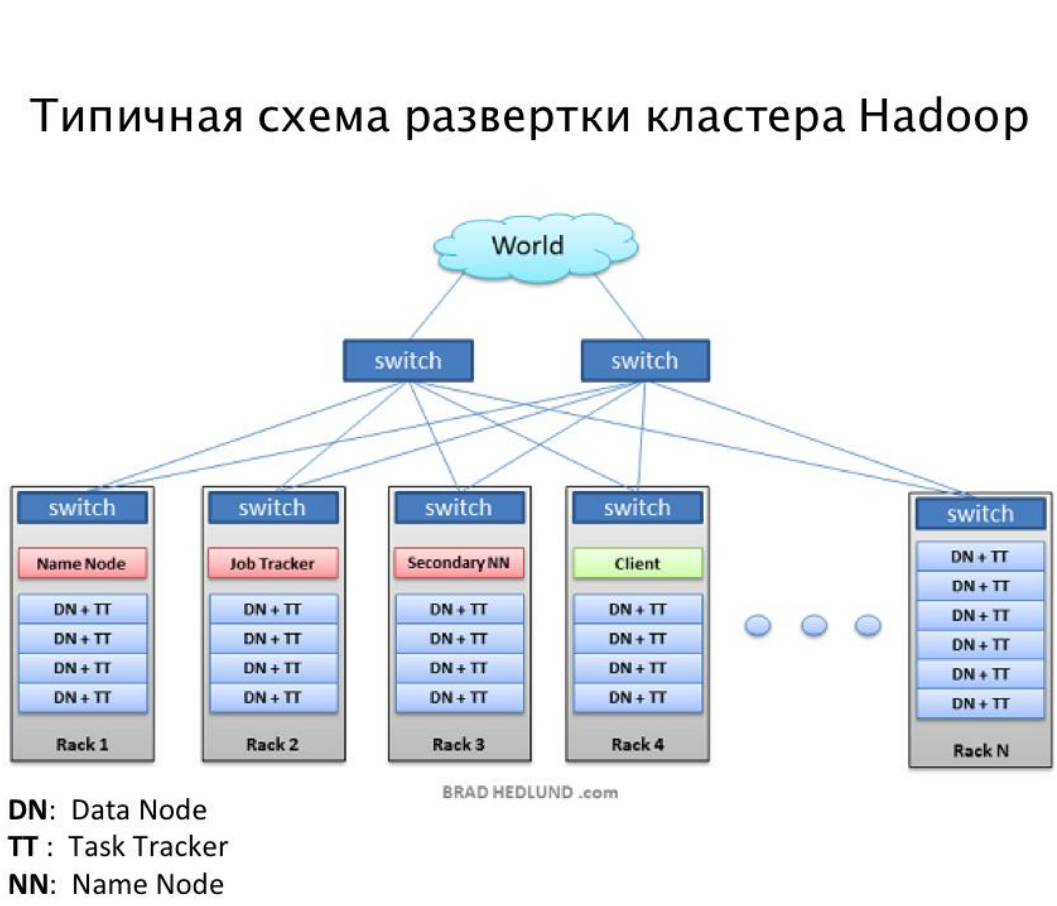

Рисунок 1. Типичная схема развертки кластера Hadoop.

В кластере Hadoop имеются следующие виды служб:

1. NameNode – основной узел, ведущий учет каждого блока каждого файла, через который производятся все операции, при отключении которого становится недоступным весь HDFS;
2. SecondaryNameNode – дополнительный узел, ведущий учет изменений (на деле не дает отказоустойчивости, потому что не заменяет собой NameNode);
3. DataNode – основные узлы кластера Hadoop, хранящие данные, ожидающие команд от главного узла Name Node;
4. NodeManager – узел системы YARN, находящийся в паре с каждым узлом DataNode, выполняющий контроль над выполнением задач с данными, хранящимися на узле;
5. ResourceManager – основной узел системы YARN, запускающий вычислительные задачи, управляющий ими и ресурсами кластера.

Все вышеперечисленные службы должны быть активны и связаны друг с другом для полноценной работы Hadoop.

Технология развертывания на одном узле кластера Hadoop

В данной работе рассматривается технология развертывания на одном физическом узле полноценного кластера Hadoop, состоящего из нескольких логических узлов - контейнеров. Рассмотрение технологии развертывания кластера проводится на конкретном примере. Главной ОС в рассматриваемом примере является Ubuntu [16].

Для создания и настройки кластера необходимо выполнить следующую последовательность действий:

1. Распределение служб Hadoop по узлам;
2. Установка виртуализации LXC;
3. Создание контейнера;
4. Установка и настройка Hadoop;
5. Клонирование контейнера;
6. Запуск кластера

Рассмотрим каждый из перечисленных шагов подробно.

Шаг 1 - распределение служб Hadoop по контейнерам

В контейнерах будут виртуализироваться такие же ОС Ubuntu. Необходимо сразу определить, как службы кластера Hadoop будут распределены по контейнерам. Распределим их по трем контейнерам следующим образом:

1. Master/Node1 контейнер содержит NameNode, SecondaryNameNode, DataNode, ResourceManager, NodeManager;
2. Node2 контейнер содержит DataNode, NodeManager;
3. Node3 контейнер содержит DataNode, NodeManager.

Итоговая архитектура (схема развертки) представлена на рис. 2.

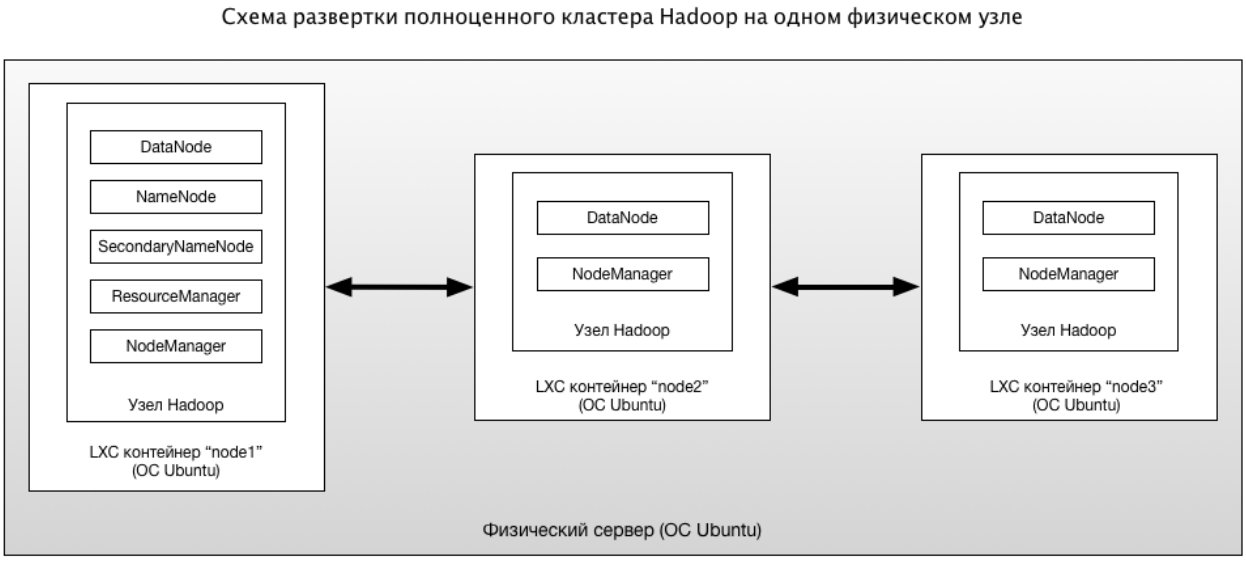

Рис. 2. Полноценный кластер Hadoop на одном физическом узле.

Такое распределение служб соответствует типовой архитектуре для реальных кластеров.

Шаг 2 - установка LXC

Перед дальнейшими действиями следует в главной ОС установить LXC. Это действие зависят от главной ОС. Для ОС Ubuntu следует выполнить следующую команду в терминале:

```
sudo apt-get install lxc
```

Для других ОС пакетный менеджер может отличаться, и может потребоваться другая команда, но в данной статье покрывается только ОС Ubuntu. Все дальнейшие действия будут выполняться в терминале (утилите командной строки).

Шаг 3 - создание LXC контейнера

При создании группы аналогичных контейнеров не стоит выполнять однообразную работу, поскольку LXC предоставляет возможность клонировать созданные контейнеры. Для создания первого контейнера выполним следующую команду:

```
sudo lxc-create –t ubuntu –n node1
```

Созданный узел будет иметь название **node1**, которое задается параметром **-n**. При необходимости, можно воспользоваться стандартным функционалом LXC для ограничения выделенных созданному контейнеру ресурсов. По аргументу **–t** передается название шаблона внутреннего контейнера. В данном случае, контейнер будет представлять собой чистую ОС Ubuntu.

После успешного создания контейнера, его следует запустить выполнением следующей команды (по параметру **–n** следует указать название контейнера, созданного ранее):

```
sudo lxc-start –n node1
```

При успешном запуске, LXC автоматически выполнит подключение к контейнеру по протоколу SSH. Потребуется войти в систему со стандартными именем пользователя **Ubuntu**, и паролем **Ubuntu**. Если, по какой-то причине, произошло отключение от запущенного контейнера, к нему всегда можно подключиться выполнив следующую команду:

```
sudo lxc-console –n node1
```

На данном этапе главная ОС поддерживает контейнер, внутри которого работает чистая, изолированная ОС Ubuntu. Следующим этапом выполним установку HDFS внутри созданного контейнера. Также в этот этап включим установку Java Development Kit, которая может отсутствовать в чистом контейнере.

Шаг 4 - установка и настройка Hadoop

Последующие действия выполняются внутри созданного контейнера с названием **node1**. Для обновления или установки JDK следует выполнить следующие действия:

```
apt-get update
```

apt-get install default-jdk

При использовании стандартного пакетного менеджера APT выполняется обновление репозиториев, а затем установка JDK.

Далее следует загрузить дистрибутив Hadoop, который в дальнейшем надо настроить. Для загрузки и размещения дистрибутива в системе следует выполнить следующие команды:

wget http://apache-mirror.rbc.ru/pub/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0-src.tar.gz

tar xfz hadoop-2.6.0.tar.gz

mv hadoop-2.6.0 /usr/local/Hadoop

Команда **wget** выполняют загрузку дистрибутива с официального сайта Apache. Затем разархивируют полученный архив (команда tar), и размещают в директории **/usr/local/Hadoop** (команде mv).

После загрузки и установки Hadoop следует его настроить. Для этого потребуется отредактировать следующие конфигурационные файлы:

1. ~/.bashrc
2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/etc/hadoop/core-site.xml
4. /usr/local/hadoop/etc/hadoop/yarn-site.xml
5. /usr/local/hadoop/etc/hadoop/mapred-site.xml
6. /usr/local/hadoop/etc/hadoop/hdfs-site.xml
7. /usr/local/hadoop/etc/hadoop/slaves

Конфигурация файла .bashrc

Конфигурационный файл **.bashrc** не относится к Hadoop, но влияет на переменные среды окружения ОС. Для работы кластера необходимо добавить следующие переменные окружения:

export JAVA_HOME=/usr/lib/jvm/java-7-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

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native

export HADOOP_OPTS=”-Djava.library.path=$HADOOP_INSTALL/lib”

Поясним перечисленные переменные:

- JAVA_HOME – путь к установленному (обновленному) JDK;
- HADOOP_INSTALL – директория, в которую был установлен Hadoop;

- PATH – содержит пути, которые проверяются на наличие исполняемых файлов, в нее требуется добавить пути к исполняемым файлам Hadoop;
- HADOOP_MAPRED_HOME – используется модулем MapReduce, указывает на директорию, в которой установлен Hadoop;
- HADOOP_COMMON_HOME – используется модулем Common;
- HADOOP_HDFS_HOME – используется HDFS;
- YARN_HOME – используется модулем YARN;
- HADOOP_COMMON_LIB_NATIVE – указывает на библиотеки и фреймворки, необходимые для работы Hadoop;

Вышеуказанные переменные команды установки переменных окружения следует добавить в конец файла **.bashrc**. При работе через утилиты Terminal и SSH это можно сделать при помощи редактора **nano** или **vi**. Команды из конфигурационного файла **.bashrc** выполняются каждый раз при создании сессии работы терминала, то есть, после сохранения указанных команд при новом подключении уже будут доступны необходимые переменные среды окружения.

Для применения переменных среды окружения в текущей сессии работы следует выполнить следующую команду:

source ~/.bashrc

Конфигурация файла hadoop-env

Файл **hadoop-env.sh** также содержит некоторые переменные окружения. В нем следует объявить переменную окружения, указывающую путь к установленному JDK. Для этого в конец файла следует добавить следующую строку:

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

Редактирование файла можно выполнить при помощи редакторов **nano** или **vi.**

Конфигурация файла core-site

В файле **core-site.xml** необходимо указать адрес **NameNode**. Так как на данный момент неизвестны адреса контейнеров, и обращаться по IP-адресу не всегда надежно, то следует задать адрес доменным именем. Затем, когда станут известны IP адреса каждого контейнера в сети, отредактируем файлы **hosts** на каждом контейнере. Файл **core-site** представляет собой XML документ, в котором требуется задать значение по ключу **fs.default.name**. Отредактированный файл выглядит примерно следующим образом:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<?xml-stylesheet type=”text/xsl” href=”configuration.xsl”?>
<configuration>
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
</property>
```

```
</configuration>
```

Таким образом, каждый узел кластера будет связан с главным узлом **NameNode**.

Конфигурация файла yarn-site

Файл **yarn-site.xml** содержит в себе все конфигурации, необходимые для выполнения задач на кластере Hadoop. В нем следует задать или отредактировать адреса службы **ResourceManager**. Все адреса указываются доменным именем. Необходимо отредактировать следующие ключи:

```
…
<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>
```

Так как в текущей схеме распределения служб по контейнерам **ResourceManager** располагается на master узле, то в адресах указываем **master**. Порты всех дочерних служб являются стандартными. При запуске ResourceManager одновременно начинает работать множество его дочерних служб, например, **yarn.resourcemanager.webapp** – веб-панель для просмотра различной информации.

Конфигурация файла mapred-site

Файл **mapred-site.xml** настраивает фреймворк для выполнения MapReduce задач на кластере, и в нем следует указать фреймворк YARN. Итоговый файл конфигурации должен выглядеть примерно так:

```
<configuration>
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
</configuration>
```

Элемент с названием **mapreduce.framework.name** должен содержать название фреймворка, поэтому указываем **yarn**.

Конфигурация файла hdfs-site

Файл **hdfs-site.xml** является одним из ключевых файлов. В нём указываются директории, в которых будут храниться данные узлов **NameNode** и **DataNode**. Еще в нем настраиваются коэффициенты репликации блоков данных, минимальный размер блока, а также множество других важных параметров, которые существенно влияют на производительность и поведение кластера при выполнении вычислительных задач. Для текущей задачи достаточно настроить коэффициент репликации и директории. Эти параметры являются необходимыми для первичной стандартной настройки и запуска кластера Hadoop.

Сперва надо создать директории, в которых узлы Hadoop будут хранить данные. Это выполняется при помощи команды **mkdir**:

```
mkdir -p /usr/local/hadoop_store/hdfs/namenode
mkdir /usr/local/hadoop_store/hdfs/datanode
```

Флаг **–p** выполнит рекурсивное создание директорий (для облегчения задачи).

Созданные директории необходимо отформатировать для работы HDFS. Вручную требуется отформатировать только директорию узла **NameNode**, а директория узла **DataNode** будет автоматически отформатирована при следующем запуске. Для этого следует выполнить следующую команду:

```
hdfs namenode –format
```

Далее следует отредактировать конфигурационный файл. Он должен иметь следующий вид (для созданных выше директорий):

```
<configuration>
<property>
        <name>dfs.replication</name>
        <value>3</value>
</property>
<property>
```

```
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

Поясним используемые элементы:

- dfs.replication – коэффициент репликации блоков данных в HDFS, обычно устанавливается равным 3;
- dfs.namenode.name.dir – путь к директории, в которой будут храниться данные NameNode;
- dfs.datanode.data.dir – путь к директории, в которой будут храниться данные DataNode.

Конфигурация файла slaves

Последний конфигурационный файл, который необходимо отредактировать – **slaves**. Этот файл используется только на основном узле кластера – **NameNode**. В нём указываются адреса всех узлов **DataNode**, которые будут использоваться в кластере. Аналогично другим адресам в конфигурационных файлах, их лучше задавать доменными именами. В текущем кластере будут использоваться следующие адреса:

node1
node2
node3

Шаг 5 - клонирование настроенного контейнера

Итак, был полностью настроен и подготовлен один контейнер. Теперь необходимо клонировать настроенный контейнер дважды. Для этого следует выйти из текущего контейнера нажатием клавиш **CTRL+D**, или же выполнив команду **exit**. Таким образом вернемся в главную ОС, где следует выполнить следующие команды:

```
sudo lxc-clone node1 node2
sudo lxc-clone node1 node3
```

В результате на основе контейнера с названием **node1** будет создано еще два контейнера – **node2** и **node3**. Их следует запустить, выполнив следующие команды:

```
sudo lxc-start –d –n node2
sudo lxc-start –d –n node3
```

Запуск контейнера с флагом **–d** отменяет автоматическое подключение к запущенному контейнеру.

Все виртуальные контейнеры по-умолчанию находятся в одной подсети. Для всех запущенных контейнеров можно получить список их IP-адресов. На каждой основной ОС автоматически выбираемая для контейнеров подсеть может быть разной. Для вывода списка контейнеров, их состояния и адреса следует выполнить следующую команду:

sudo lxc-ls –f

Флагом **–f** задается требование вывести все контейнеры, включая выключенные.

В консоль будет выведен список, пример которого приведен на рисунке 3.

```
ravis@UbuntuServer:~$ sudo lxc-ls -f
[sudo] password for ravis:
NAME   STATE    IPV4         IPV6  AUTOSTART
----------------------------------------------
node1  RUNNING  10.0.2.195   -     NO
node2  RUNNING  10.0.2.12    -     NO
node3  STOPPED  -            -     NO
ravis@UbuntuServer:~$
```

Рисунок 3. Пример вывода списка LXC контейнеров.

В примере на рисунке 3 контейнер **node3** находится в остановленном состоянии. В данном случае все три контейнера должны иметь состояние **RUNNING**, и должны быть доступны IPv4 адреса.

Далее следует войти в каждый контейнер при помощи следующей команды:

sudo lxc-console –n node1

Затем выполнить настройки файла **hosts** на каждом логическом узле. Файл находится в директории **/etc/** и не имеет расширения. Этот файл проверяется в первую очередь при выполнении любого исходящего запроса в сеть. Если находится совпадение по доменному имени в этом файле, то из него берется указанный для доменного имени IP-адрес, и на него отправляется запрос. Соответственно, при текущем распределении служб и приведенном выше примере файл **hosts** должен содержать следующие пары IP-домен:

127.0.0.1 localhost
10.0.2.195 master
10.0.2.195 node1
10.0.2.12 node2
10.0.2.XXX node3

Таким образом, Hadoop будет использовать доменные имена, которые внутри каждого контейнера будут преобразовываться в правильные IP-адреса.

Шаг 6 -запуск кластера

Запуск кластера выполняется с контейнера, в котором располагается **NameNode**. В данном случае это контейнер с названием **node1.** Hadoop запускается в два этапа:

1. Запуск файловой системы HDFS – включаются службы NameNode, SecondaryNameNode, DataNode;
2. Запуск YARN – включаются ResourceManager, NodeManager.

Запуск **slave** узлов кластера выполняется автоматически **master** узлом, через протокол SSH. Для запуска HDFS следует выполнить следующую команду на контейнере **node1**:

start-dfs.sh

В результате будут запущены все DataNode узлы и в консоли появятся соответствующие сообщения.

Для проверки результата запуска следует выполнить команду **jps**, которая выводит все Java процессы в системе. Вывод на контейнере с названием node1 должен быть следующим:

XXX NameNode
XXX SecondaryNameNode
XXX DataNode
XXX Jps

Одной из ключевых особенностей HDFS является автоматическая репликация блоков данных. Допустим, один из узлов многотысячного кластера станет недоступен. В этом случае, при коэффициенте репликации равном 3, каждый блок данных недоступного узла будет доступен на еще двух узлах кластера, и работа будет продолжена.

На **slave** узлах кластера, то есть контейнерах **node2** и **node3** должны работать следующие процессы:

XXX DataNode
XXX Jps

На данном этапе можно сохранять и манипулировать данными и файлами в HDFS. Но для выполнения задач необходимо запустить YARN. Для этого следует выполнить следующую команду:

start-yarn.sh

Аналогично HDFS, эта команда запустит необходимые процессы на всех **slave** узлах автоматически. В результате на каждом узле в списке Java процессов должен дополнительно появиться процесс **NodeManager**, а на узле **node1**, на который был ранее распределен **ResourceManager,** должен появиться одноименный процесс.

Заключение

В результате практической апробации предлагаемой технологии был успешно настроен и запущен полноценный кластер Hadoop в рамках одного физического узла. Без методов виртуализации возможен запуск Hadoop только в режимах Single-Node или Pseu-

do-Cluster. Эти режимы не дают возможности полноценно протестировать кластер для оценки эффективности задач, которые необходимо на нём решить. Предложенная технология позволяет настроить и запустить полноценный кластер и тем самым создать экономичный тестовый стенд для разработчиков распределенных приложений.

При должном подходе к автоматизации создания контейнеров из созданных заранее шаблонов, а также автоматически настраивая сетевые маршруты между контейнерами и сетью Internet, можно реализовать облачную IaaS-/PaaS-платформу. Внутренний функционал контейнеров не ограничивается Hadoop, в них может быть реализовано что угодно, включая клиентские задачи.

Аналогичный подход используется у некоторых современных облачных провайдеров. Например, PaaS-платформа Heroku использует LXC контейнеры для создания так называемых Dyno, в которых запускаются любые серверные приложения клиента. Количество одинаковых Dyno можно в любой момент времени менять, что позволяет гибко распоряжаться ресурсами. С точки зрения реализации, Dyno – это обычный LXC контейнер, который соответствующим образом выведен в сеть Internet. Серверные приложения клиентов попадают в Heroku при выполнении загрузки кода в git-репозиторий, предоставляемый платформой.

Облачная платформа Rackspace использует CoreOS, которая базируется на Docker [17,18], который представляет собой модифицированный вид контейнерной виртуализации.

Таким образом, предложенная технология может быть использована для практической реализации облачных вычислений.

Список литературы

1. Горячкин Б.С., Демьянов В.Л., Ященко О.С. Концептуальные основы развития интернет-портала МГТУ им. Н.Э. Баумана // Инженерный журнал: наука и инновации. / Электронное научно-техническое издание. 2012. № 3 (3). С. 46-53. Режим доступа: http://engjournal.ru/catalog/it/asu/103.html (дата обращения: 28.02.2015)
2. Багдасарян Е.А., Королев В.С., Черненький М.В. Архитектура системы управления рабочим процессом // Инженерный вестник: электронный научно-технический журнал МГТУ им. Н.Э. Баумана. 2013. № 10. С. 519 - 526. Режим доступа: http://engbul.bmstu.ru/doc/649649.html (дата обращения: 28.02.2015)
3. Виноградова М.В., Белоусова В.И., Мельник В.Н. Концепции разработки и организации системы прозрачного доступа к файлам больших объемов и raid-массивам в распределенной разнородной среде // Инженерный вестник: электронный научно-технический журнал МГТУ им. Н.Э. Баумана. 2013. № 9. С. 549 - 560. Режим доступа: http://engbul.bmstu.ru/doc/637968.html (дата обращения 28.02.2015).
4. Черненький В.М., Гапанюк Ю.Е. Методика идентификации пассажира по установочным данным. // Инженерный журнал: наука и инновации. / Электронное научно-техническое издание. 2012. № 3(3). С. 30 – 39. Режим доступа: http://engjournal.ru/catalog/it/biometric/89.html (дата обращения 28.02.2015).

5. Виноградов В.И., Мазнев В.Г. Метод поиска часто повторяющихся маршрутов в пространственно-временных данных // Инженерный журнал: наука и инновации. / Электронное научно-техническое издание. 2012. № 3 (3). С. 13 - 19. Режим доступа: http://engjournal.ru/catalog/it/asu/100.html (дата обращения 28.02.2015).
6. Постников В.М., Спиридонов С.Б. Подход к расчету весовых коэффициентов ранговых оценок экспертов при выборе варианта развития информационной системы // Наука и образование: электронное научно-техническое издание МГТУ им. Н.Э. Баумана. 2013. № 8. С. 395 - 412. **DOI:** 10.7463/0813.0580272 Режим доступа: http://technomag.bmstu.ru/doc/580272.html (дата обращения 28.02.2015).
7. Галкин В.А., Осипов А.В. Оценка параметров системы мониторинга рабочих станций в локальной вычислительной сети // Инженерный вестник: электронный научно-технический журнал МГТУ им. Н.Э. Баумана. 2014. № 10. С. 528 - 539. Режим доступа: http://engbul.bmstu.ru/doc/730565.html (дата обращения 28.03.2015).
8. Тоноян С.А., Тимофеев В.Б., Черненький С.В. Анализ и выбор конфигурации сети для финансово-экономической деятельности МГТУ им. Н.Э. Баумана на базе платформы «1С: Предприятие 8» // Инженерный журнал: наука и инновации. / Электронное научно-техническое издание. 2012. № 3 (3). С. 101 - 108. Режим доступа: http://engjournal.ru/catalog/it/asu/110.html (дата обращения 28.03.2015).
9. Lublinsky B., Smith K.T., Yakubovich A. Professional Hadoop Solutions. United States: Wrox Press, 2013. 504 p.
10. Hadoop: Официальный веб-сайт. Режим доступа: http://hadoop.apache.org/ (дата обращения 1.04.2015).
11. White T. Hadoop: The Definitive Guide, 4th Edition. O'Reilly Media, Incorporated. 2015. 768 p.
12. Schulz G. Cloud and virtual data storage networking: your journey to efficient and effective information services. CRC Press - Taylor & Francis Group, 2011. 370 p.
13. Strauss D. Containers—Not Virtual Machines—Are the Future Cloud // Linux Jurnal. 2013. 17 Jul. Режим доступа: http://www.linuxjournal.com/content/containers%E2%80%94not-virtual-machines%E2%80%94are-future-cloud (дата обращения 1.04.2015).
14. LXC / Introduction // Linux Containers. Режим доступа: https://linuxcontainers.org/lxc/introduction/ (дата обращения 1.04.2015).
15. LXC / Documentation (Документация по API LXC). // Linux Containers. Режим доступа: https://linuxcontainers.org/lxc/documentation/ (дата обращения 1.04.2015).
16. Виртуальный Linux // IBM developerWorks Россия. Режим доступа: https://www.ibm.com/developerworks/ru/library/l-linuxvirt/ (дата обращения 1.04.2015).
17. Turnbull J. The Docker Book. / James Turnbull, 2015. Version: v1.5.0. 57 p. Режим доступа: http://dockerbook.com/TheDockerBook_sample.pdf (дата обращения 1.04.2015).
18. Docker / Официальный веб - сайт. Режим доступа: https://www.docker.com/ (дата обращения 1.04.2015).