ТЕМА 7.2 ИНТЕРНЕТ ВЕЩЕЙ И ТЕХНОЛОГИИ РАБОТЫ С BIG DATA

Автор: к.т.н., доцент ВШПИ СПбПУ Никифоров И.В

УПРАЖНЕНИЕ 1 – УСТАНОВКА HADOOP И РАБОТА С HDFS

Hadoop[1] — проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из нескольких сотен или тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоко нагруженных веб-сайтов. Разработан на языке Java в рамках вычислительной парадигмы MapReduce[2], согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера (компьютерах) и естественным образом сводимых в конечный результат.

Основной операционной системой Hadoop является Linux, но возможна установка и на операционной системе Windows. Для общности и простоты разбирательства с Hadoop рекомендуется использовать Linux. Для тех, у кого эта система отсутствует и основная операционная система Windows, предлагается использовать VirtualBox с установленной клиентской ОС Linux или любой другой способ виртуализации.

Для успешного выполнения упражнения необходимо умение пользоваться документацией на используемые технологии.

Для выполнения упражнений необходимо осуществить предварительную конфигурацию рабочего окружения. Ниже перечисленные основные шаги.

1. Установите Linux в Virtual Box и настройте общую папку для того, чтобы из домашней операционной системы было проще и привычнее переносить файлы. Рекомендуется использовать Ubuntu не ниже 14.0 версии.

2. Установите дополнения гостевой ОС для Virtual Box.

3. Установите JDK. Минимальная версия Java 1.7, но рекомендуемая Java 1.8 (поддерживается, начиная с версии Hadoop 2.3).

На момент создания упражнений вышла версия Hadoop 3.0[3], но она еще не является стабильной. Исходя из этого рекомендуется использовать версию 2.x.x.

Цель данного упражнения — получение практических навыков установки и администрирования фреймворка Apache Hadoop и распределённой системой HDFS[1].

---

[1] Tom White. Hadoop: The Definitive Guide, 4th Edition, O'Reilly Media, 2015.

[2] Thilina Gunarathne. Hadoop MapReduce Cookbook. Hadoop MapReduce v2 Cookbook Second Edition. Packt Publishing, 2015

[3] Официальный сайт Apache Hadoop, предоставляющий возможность скачать любую версию фреймворка. [Электронный ресурс]. Режим доступа: // http://www.apache.org/dyn/closer.cgi/hadoop/common/

Упражнение затрагивает установку и запуск Hadoop, работу с файловой системой и с веб-интерфейсом фреймворка.

Теоретический материал

Одной из основных и важных характеристик фреймворка Apache Hadoop является работа с распределенной моделью данных и вычислений. Данный фреймворк рассчитан на работу на кластере. На каждом компьютере кластера должно присутствовать две компоненты: Task Tracker[2] и Data Node[3]. Task Tracker — выполняет маленькую вычислительную задачу на узле. Задача Data Node — управлять данными, которые были переданы именно этому узлу. Узлы, которые содержат только две данных компоненты, носят название Slave. Также в системе должен присутствовать Master узел, координирующий работу Slave узлов. Master узел имеет в себе не только компоненты Task Tracker и Data Node, но и Job Tracker[4] и Name Node[5].

Job Tracker и Task Tracker — это части компоненты MapReduce.

Name Node и Data Node — это части компоненты HDFS.

Приложение, которое будет исполняться на Hadoop, будет контактировать только с Master Node. Основная архитектура приложения — это набор процессов.

Роль и цель Job Tracker, который исполняется на Master Node, — разбить большие задачи на маленькие кусочки и отправить каждую маленькую часть вычислений соответствующему Task Tracker на узле. Task Tracker в свою очередь, когда сможет закончить вычисления, отправит обратно информацию в Job Tracker, который объединит все результаты вычислений и отправит их обратно приложению.

Цель Name Node — хранить индекс того, какая часть данных находится на каком узле. Приложение обращается к Name Node и получает от него информацию об узле, где хранится кусок данных и затем обращается к нужному узлу напрямую.

Обобщенная схема архитектуры фреймворка Apache Hadoop представлена на рис. 1.

---

[1] Пользовательское руководство по HDFS от Apache. [Электронный ресурс]. Режим доступа: // https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

[2] Документация Hadoop по TaskTracker. [Электронный ресурс]. Режим доступа: // https://wiki.apache.org/hadoop/TaskTracker

[3] Документация Hadoop по DataNode. [Электронный ресурс]. Режим доступа: // https://wiki.apache.org/hadoop/DataNode

[4] Документация Hadoop по JobTracker. [Электронный ресурс]. Режим доступа: // https://wiki.apache.org/hadoop/JobTracker

[5] Документация Hadoop по NameNode. [Электронный ресурс]. Режим доступа: // https://wiki.apache.org/hadoop/NameNode

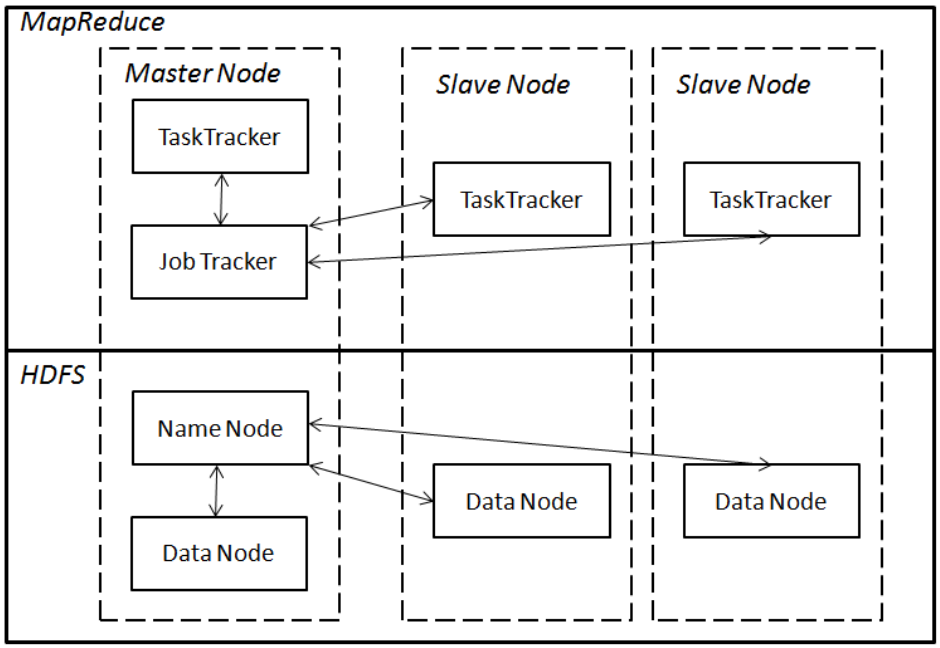

Рис. 1. Архитектура фреймворка Apache Hadoop

Существует несколько вариантов использования Hadoop:

— автономный режим (Standalone Mode) используется для того, чтобы на локальной машине применить Hadoop как кластер, состоящий из одного узла. Данные хранятся в локальной файловой системе;

— псевдо распределенный режим (Pseudo Distributed Mode). В данном режиме Hadoop работает в «вырожденном кластере». Все задачи выполняются на одной машине отдельными процессами, для хранения данных используется HDFS, так же используется JobTracker;

— полностью распределенный режим (Fully Distributed Mode) используется на реальном кластере.

Для каждого из вариантов требуется своя последовательность установки и настройки. Для выполнения упражнения предлагается установить Hadoop в псевдораспределённом режиме для возможности работы с HDFS.

Ход выполнения упражнения

1. Установите Apache Hadoop на компьютер в псевдо распределённом режиме. Для работы с Hadoop создайте отдельного пользователя системы. Например, назовите пользователя — hduser.

2. Запустите сервисы Hadoop.

3. Используя команду *jps*[1] проверьте, что успешно запущены все демоны Hadoop: NameNode, DataNode, NodeManager[2] и ResourceManager[3].

4. Изучите команды для работы с HDFS[4]:

— навигация по директориям системы и отображение их содержимого;

— копирование из локальной системы в HDFS и обратно;

— вывод содержимого файла;

— удаление файла/директории;

— и др.

5. Создайте любой текстовый файл в домашней директории, скопируйте его в HDFS (для сдачи упражнения, необходимо показать 2 способа помещения файла в HDFS)

6. Из HDFS выведите в консоль содержимое скопированного файла.

7. Скопируйте файл назад в домашнюю директорию.

8. Откройте в браузере веб-интерфейс Hadoop, доступный по ссылке: http://localhost:50070 (адрес - по умолчанию), и посмотрите с помощью веб-интерфеса содержимое файловой системы.

Для успешной защиты упражнения требуется:

— запустить Hadoop у себя на компьютере;

— с помощью команды *jps* показать, что демоны (NameNode, DataNode, NodeManager, ResourceManager) успешно запущены;

— выполнить все шаги, связанные с работой с HDFS;

— запустить веб-интерфейс Hadoop и показать в нём созданные в HDFS файлы.

УПРАЖНЕНИЕ 2 – УСТАНОВКА SPARK И РАБОТА С SPARK SHELL

Spark[5] — фреймворк, являющийся одним из проектов фонда Apache Software Foundation, который представляет собой набор open-source библиотек для разработки приложений распределённой обработки неструктурированных и слабоструктурированных данных. Apache Spark входит в экосистему проектов Hadoop. Apache Spark использует

---

[1] Документация Oracle по jps – команде, отображающей Java-процессы. [Электронный ресурс]. Режим доступа: // http://docs.oracle.com/javase/7/docs/technotes/tools/share/jps.html

[2] Документация Hadoop по NodeManager. [Электронный ресурс]. Режим доступа: // http://hortonworks.com/blog/apache-hadoop-yarn-nodemanager/,

[3] Документация Hadoop по ResourceManager. [Электронный ресурс]. Режим доступа: // http://hortonworks.com/blog/apache-hadoop-yarn-resourcemanager/

[4] Список команд HDFS. [Электронный ресурс]. Режим доступа: // http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

[5] Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills. Advanced Analytics with Spark, *O'Reilly Media*, 2015.

примитивы данных для обработки непосредственно в оперативной памяти. За счет такой in-memory обработки достигается значительный выигрыш во времени исполнения по сравнению с традиционной концепцией обработки данных Hadoop MapReduce для итеративных алгоритмов, которым требуется постоянное обращение к дисковому пространству.

Несмотря на то, что Apace Spark изначально разработан на языке Scala, он предоставляет широкий API для разработки приложений на различных языках программирования, отличных от родного: Java, Python, R

На сегодняшний день актуальной версий Apache Spark является версия 2.1.0.

В состав проекта Spark входят следующие компоненты, перечисленные ниже.

— Spark Core – реализация основных функциональных возможностей, к которым относятся: исполнение программ, планирование, обработка исключительных ситуаций и восстановление после ошибок, логирование, взаимодействие с источниками данных, управление и распределение памяти и др.

— Spark Streaming – компонент для обработки данных в потоковом режиме. Входные данные могут поступать с различных брокеров сообщений, таких как Apache Kafka, Flume или RabbitMQ, а также через предоставляемый им программные интерфейсы.

— MLlib – компонент, содержащий реализацию таких методов машинного обучения, как: классификация, регрессия, кластеризация, и др.. Отличительной особенностью этого компонента является то, что все методы способны работать в кластерной инфраструктуре и позволяют осуществлять гибкую горизонтальную масштабируемость.

— Spark SQL – компонент, позволяющих осуществлять работу со структурированными данными за счет разработки и исполнения SQL-запросов, в том числе и поддержке его диалекта Hive Query Language для Hadoop. В дополнении, компонент позволяет осуществлять перемешивание SQL-кода с кодом на таких языках как Java, Scala, Python, которые поддерживаются абстракциями RDD.

— GraphX – компонент-библиотека, поддерживающая параллельную обработку графов.

Приложения, созданные с помощью фреймворка Apache Spark могут быть запущены в одном из следующий режимов: полностью автономный режим (локальный), режим работы на кластере под управлением ресурс менеджеров Yarn или Mesos.

Цель данного упражнения – получение практических навыков установки и администрирования фреймворка Apache Spark. Упражнение покрывает установку и запуск Spark, работу с командными строками и с веб-интерфейсом фреймворка.

Теоретический материал

Приложения, созданные при помощи Apache Spark, запускаются как набор независимых процессов на кластере, управляемые объектом SparkContext[1] в основной программе, которая называется Program Driver. Для запуска на кластере SparkContext может подключаться к различным менеджерам ресурсов. При запуске а автономном режиме необходимо создавать SparkContet вручную. После подключения Spark может обращаться к исполнителям – процессам на узлах кластера, которые выполняют операции вычисления и хранения данных для основного приложения. Затем, он отправляет код приложения и задачи исполнителям. Обобщенная схема архитектуры Spark представлена на рис. 2

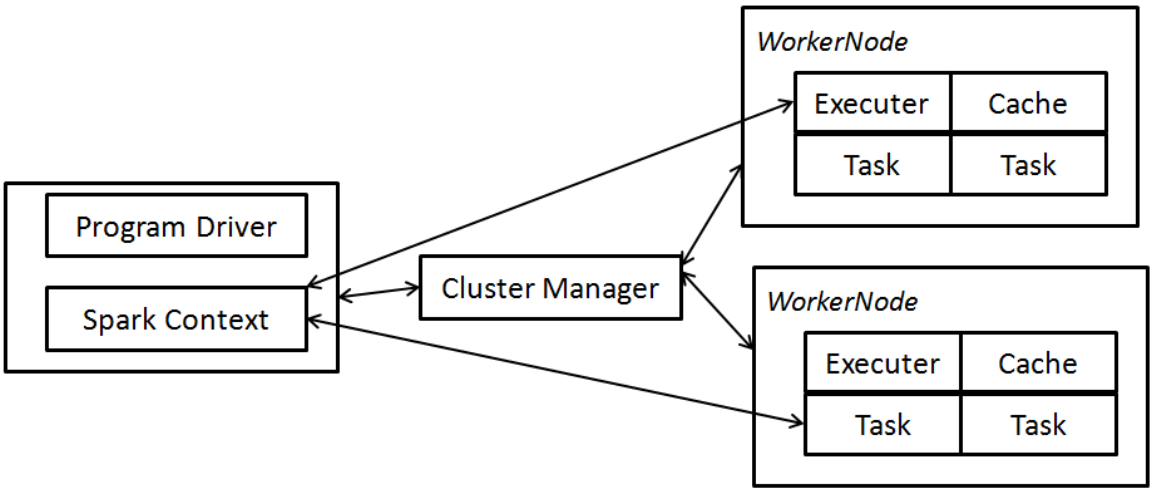

Рис. 2. Архитектура Spark

В представленной архитектуре стоит отметить несколько деталей:

— каждое приложение получает своего отдельного исполнителя, которое привязывается к нему на всё время исполнения, и выполняет задачи в несколько потоков. Это имеет определённые преимущества, но также означает, что данные не могут быть совместно использованы несколькими приложениями (одно приложение соответствует одной сущности SparkContext), без записи их на какое-либо внешнее хранилище;

— для Spark не имеет значения используемый менеджер ресурсов. Это может быть YARN, Mesos или любой другой менеджер;

— т.к. Program Driver планирует и запускает задачи на кластере, не следует использовать рабочие узлы «далеко» от основного, желательно иметь их в одной локальной сети.

Основная парадигма распределенных вычислений над большими данными заключается в том, что необходимо вычисления перенести к данным, а не данные к вычислениями, т.к. последнее крайне затратная операция.

---

[1] Документация по Spark Context. [Электронный ресурс]. Режим доступа: // https://spark.apache.org/docs/1.4.0/api/java/org/apache/spark/SparkContext.html

Ход выполнения упражнения

1. Установите Apache Spark на локальный компьютер. Для работы со Spark рекомендуется, также как и в упражнении по Hadoop, создать отдельного пользователя системы. Например, назовите пользователя – sparkuser. Обратите внимание на то, что для установки Spark требуется установить sbt.

2. Используя команду *pyspark*, запустите командную строку Spark, используя язык Python, проделайте базовые операции, например, выведите на печать «Welcome to Spark».

3. Используя команду *spark-shell*, запустите командную строку Spark, используя язык Scala, проделайте базовые операции, как и в предыдущем пункте.

4. Откройте в браузере веб-интерфейс Spark, доступный по ссылке: http://localhost:4040 (адрес по-умолчанию), и изучите его возможности. Веб-интерфейс доступен, когда запущена командная строка.

Для успешной сдачи упражнения требуется:

— запустить Spark на локальном компьютере;

— запустить командную строку (любую) и вывести в ней на печать «Welcome to Spark»;

— запустить веб-интерфейс Spark.