ISSN 1814-1196
Научный вестник НГТУ
том 55, № 2, 2014, с. 97–104

http://journals.nstu.ru/vestnik
Scientific Bulletin of NSTU
Vol. 55, No. 2, 2014, pp. 97–104

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

MODERN INFORMATION TECHNOLOGIES

УДК 004.75

Особенности развёртывания, настройки и применения инструментария apache hadoop на windows и unix-подобных операционных системах*

И.А. БОТЫГИН[1], А.В. ЗАБЕЙВОРОТА[2]

[1] 634050, г. Томск, проспект Ленина, 30, Национальный исследовательский Томский политехнический университет, к. т. н., доцент, e-mail: bia@tpu.ru
[2] 634050, г. Томск, проспект Ленина, 30, Национальный исследовательский Томский политехнический университет, магистрант, e-mail: ghmulti@gmail.com

Актуальность работы обусловлена необходимостью обработки больших объемов разнородных данных. **Цель работы:** выявить особенности настройки и применения свободно распространяемого инструментария проектирования распределенных систем для хранения, анализа и обработки данных на базе проекта Apache Software Foundation – Apache Hadoop. **Методы исследования:** экспериментальный анализ отладки и тестирования разработанных программных кодов в средах специализированных фреймворков. **Результаты:** описаны алгоритмические схемы формирования инфраструктур, необходимых для функционирования Apache Hadoop, и процесс настройки Apache Hadoop для ОС Windows (Cygwin) и Ubuntu. Процесс управления осуществлялся через FS shell-интерпретатор, запускаемый из консоли операционной системы. Исследована базовая схема взаимодействия компонентов архитектуры Apache Hadoop при организации распределенной обработки данных. Показана возможность формирования кластера компьютеров с неограниченным горизонтальным масштабированием и параллельным выполнением заданий. Проведено сравнение способа конфигурирования приложений на основе Java-классов и подхода на основе xml-конфигураций с использованием Spring Hadoop-фреймворка, который комбинирует возможности Spring Framework с возможностью Apache Hadoop. Показывается возможность платформы Spring Hadoop обеспечить слабую связанность компонентов и поддержку всестороннего доступа к данным HDFS, тем самым делая решение более гибким и модульным. Предложена и апробирована технологическая схема создания приложения, реализующего парадигму MapReduce.

Ключевые слова: Apache Hadoop, фреймворк Spring Hadoop, MapReduce-вычисления, параллельные вычисления, хранилища данных, масштабируемость, распределенная файловая система Hadoop, большие данные, управление кластером Apache Hadoop, мониторинг кластера Apache Hadoop, развертывание кластера Apache Hadoop

ВВЕДЕНИЕ

В настоящее время получают широкое распространение системы распределенного хранения и обработки больших объемов данных [1, 2]. Такие системы находят свое применение не только в наукоемких областях, но и при обработке и анализе данных информационных ресурсов глобальных телекоммуникаций [3, 4]. Не вдаваясь в анализ существующих решений в области распределенных вычислений, отметим проект Apache Hadoop фонда Apache Software Foundation, получивший наибольшее распространение и популярность среди крупнейших игроков IT рынка. Основными функциональными компонентами Apache Hadoop являются Hadoop Common (библиотеки управления файловыми системами), Hadoop Distributed File System (распределенная файловая система), Hadoop YARN (планирование вычислениями и

* *Статья получена 12 февраля 2014 г.*

ресурсами кластера), Hadoop MapReduce (набор классов для выполнения распределённых вычислений в рамках парадигмы MapReduce) [5].

Необходимо заметить, что использование инструментария Apache Hadoop требует серьезной профессиональной подготовки и компетенций в области сетевых технологий и информационных коммуникаций, хранилищ данных, параллельных вычислений. Но, вместе с тем, не следует упускать и такой важный момент, как целесообразность использования Apache Hadoop. Эффект от применения Apache Hadoop проявляется только при решении действительно трудоемких задач, требующих одновременно и большого объема вычисления и хранения больших объемов данных. Для небольших вычислительных задач трудозатраты по их развертыванию в среде Apache Hadoop могут привести к неоправданно высокой стоимости полученных результатов.

1. НАСТРОЙКА ОКРУЖЕНИЯ ДЛЯ ПЛАТФОРМ WINDOWS И UNIX

Apache Hadoop реализован с использованием технологии Java, поэтому на всех компьютерах, планируемых для размещения в кластере вычислений, необходима установка виртуальной машины Java Runtime Environment (JRE). Минимальная реализация JRE, необходимая для исполнения Java-приложений, доступна для бесплатного использования на официальном сайте фирмы Oracle [6]. Для разработки собственного приложения, работающего на проектируемой распределенной системе (кластере), необходимо установить Java Development Kit (JDK).

Обязательным условием также является установка программного обеспечения, позволяющего устанавливать Secure Shell (SSH) соединения, необходимого для шифрования данных при взаимодействии между всеми узлами проектируемой распределенной системы.

Настройка окружения для Windows. Для развертывания приложения под ОС Windows необходимо установить Cygwin [7], обеспечивающего интеграцию Windows-приложений и данных с приложениями и данными UNIX-подобной среды. При установке требуется включить поддержку сетевого протокола прикладного уровня SSH и произвести его настройку.

Шаг 1. Запустить Cygwin от имени администратора и ввести команду ssh-host-config, которая сгенерирует конфигурационные файлы и выдаст запрос на подтверждение разделения доступа по привилегиям. Рекомендуется ответить согласием (yes).

Шаг 2. Создать учетную запись для SSH с особыми привилегиями.

Шаг 3. Сформировать условие запуска процесса для приема SSH-соединений как службу, что позволит получить SSH-доступ независимо от того, запущен ли в данный момент Cygwin.

Шаг 4. Создать привилегированный аккаунт с именем пользователя (по умолчанию формируется имя – cyg_server).

Шаг 5. Сформировать пароль для привилегированного аккаунта.

Шаг 6. Ввести команду для запуска службы *sshd*: *net start sshd*.

Шаг 7. Выйти из интерфейса Cygwin (exit).

Шаг 8. Создать SSH-ключи привилегированного аккаунта (в консоли необходимо ввести команду *ssh-user-config*). Рекомендуется настроиться на схему SSH2 (более безопасна и предпочтительнее для хранения ключей).

Шаг 9. Создание SSH2_DSA_ID-файла, если необходимо получать доступ не по паролю, а по файлу-ключу.

Шаг 10. Для проверки конфигурации ввести в консоли Cygwin команду ssh –v localhost. Ключ –v включит режим *verbose*, который покажет все детали процесса. При этом необходимо ввести пароль, а если имя пользователя по умолчанию было изменено, то необходимо ввести и его. В случае успеха после выполнения команды будет отображена стандартная bash-строка.

Настройка окружения для Unix. Для запуска Apache Hadoop под UNIX-системой нет необходимости устанавливать дополнительные приложения, но настройка SSH все равно необходима.

Шаг 1. Создание пользователя:

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
```

После выполнения скрипта пользователь hduser будет добавлен в группу hadoop на локальном компьютере.

Шаг 2. Создать SSH-ключ с заданным паролем:

su – hduser

ssh-keygen -t rsa -P ""

Шаг 3. Добавить созданный ключ в авторизованные ключи:

cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

После завершения настроек окружения необходимо загрузить и распаковать собственно Apache Hadoop (библиотеки для разработки распределенных приложений). Все конфигурационные файлы Apache Hadoop находятся в папке conf. В файле hadoop-env.sh необходимо указать расположение JRE или JDK. Файл conf/core-site.xml описывает конфигурацию файловой системы проектируемой распределенной системы (место, где будут храниться обрабатываемые файлы). Файл conf/mapred-site.xml описывает один или множество процессов управления и координирования кластером вычислителей, которые будут выполнять необходимые задания. Файл conf/hdfs-site.xml представляет описание синхронизации содержимого нескольких копий файлов (репликации).

2. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ЗАПУСКА НА КЛАСТЕРЕ ВЫЧИСЛИТЕЛЕЙ

Задачи, решаемые с помощью Hadoop MapReduce, должны отвечать одному основному требованию – они должны относиться к классу задач, допускающих распараллеливание потока данных для обработки. В качестве примера реализуем приложение для поиска наиболее часто встречающихся слов в текстовых файлах.

Функциональная структура кластера вычислителей для такой обработки представлена на рис. 1.

Ошибка!

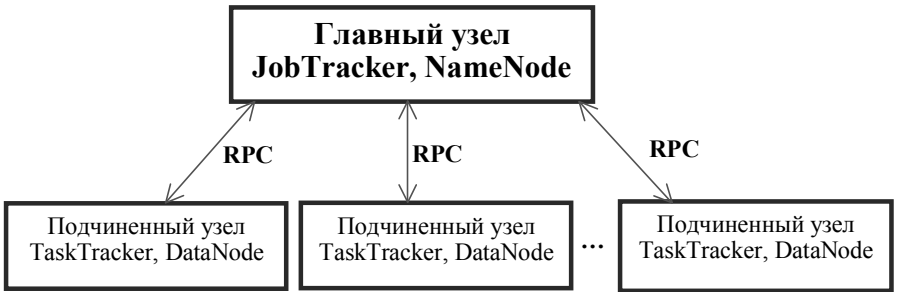

Рис. 1. Схема взаимодействия процессов в кластере

Подчиненные узлы получают задания от главного узла, данные для обработки распределяются также между подчиненными узлами, главный узел выступает в роли координатора. На главном узле работают процессы, отвечающие за управление и координирование кластером вычислителей (JobTracker, NameNode), на подчиненных узлах работают процессы, обеспечивающие функции хранения данных в файловой системе (HDFS) и реализующие функционал MapReduce (TaskTracker, DataNode). JobTracker распределяет задачи между узлами DataNode, а TaskTracker выполняет полученные задания. Кроме того, JobTracker проверяет результаты выполнения назначенных заданий, и, если один из узлов по какой-то причине выходит из строя, незавершенное задание переназначается на другой узел. NameNode является главным сервером Hadoop и управляет пространством имен файловой системы, а также доступом к файлам, хранящимся в кластере вычислителей. Взаимодействие TaskTracker-узлов с узлом JobTracker идет посредством RPC-вызовов, причем вызовы идут только от TaskTracker. Аналогичный принцип взаимодействия реализован в HDFS – между узлами DataNode и NameNode-узлом. Такое решение позволяет уменьшить зависимость управляющего процесса от управляемых.

Решение задачи сводится к реализации обработчика данных, который на каждом вычислительном узле будет выполнять параллельную обработку входных данных и формирование промежуточных результатов работы (класс, реализующий интерфейс Mapper). Также должен быть реализован обработчик, который обрабатывает полученные данные с узлов (класс реализующий интерфейс Reducer).

Для загрузки папки с исходными файлами в файловую систему HDFS выполним:

hadoop dfs -copyFromLocal /localfolder /input

Все данные из файлов будут построчно считаны и переданы объектам класса-обработчика – назовем его TopWordMapper. Объект класса TopWordMapper обеспечит преобразование переданной ему строки в пары ключ-значение для каждого слова, где само слово будет являться ключом, а значением будет единица. На каждом подчиненном узле будет работать свой экземпляр TopWordMapper. После того как все строки будут разобраны, пары ключ-значение группируются в списки по ключу и передаются на вход второго класса-обработчика – назовем его TopWordReducer. Объект класса TopWordReducer обеспечит подсчет общего количества вхождений конкретного слова и, при необходимости, включит его в результат работы приложения. Результатом работы будут множество пар ключ-значение, где ключ – слово, а значение – количество вхождений данного слова.

TopWordMapper. Класс TopWordMapper должен быть унаследован от класса org.apache.hadoop.mapreduce.Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>.

Параметры класса Mapper описывают:

– KEYIN – тип ключа, подаваемого на вход классу Mapper;

– VALUEIN – тип значения, подаваемого на вход классу Mapper;

– KEYOUT – тип выходного ключа класса Mapper;

– VALUEOUT – тип выходного значения класса Mapper.

Переопределим метод map(KEYIN key, VALUEIN value, Mapper.Context context) класса TopWordMapper для реализации необходимой нам функциональности. В нем будет выполняться следующая логика.

1. Разбиение входных значений на слова:

StringTokenizer tokenizer = new StringTokenizer(value.toString());

2. Создание выходных пар ключ-значение (метод write у класса Mapper.Context с необходимыми параметрами):

context.write(tokenizer.nextToken(), new IntWritable(1));

Исходный код класса TopWordMapper:

```
package com.sample;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class TopWordMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(value.toString());
            while (tokenizer.hasMoreTokens()) {
                    word.set(tokenizer.nextToken());
                    context.write(word, one);
            }
    }
}
```

TopWordReducer. Класс TopWordReducer должен быть унаследован от класса org.apache.hadoop.mapreduce.Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>.

Параметры класса Reducer описывают:

– KEYIN – тип ключа, подаваемого на вход классу Reducer. Это значение не будет совпадать со значением KEYIN для класса Mapper;

– VALUEIN – тип значения, подаваемого на вход классу Reducer. Это значение не будет совпадать со значением VALUEIN для класса Mapper;

– KEYOUT – тип выходного ключа класса Reducer;

– VALUEOUT – тип выходного значения класса Reducer.

Переопределим метод public void reduce(KEYIN key, Iterable<VALUEIN> values, Context context) для реализации необходимой нам функциональности. В нем будет выполняться следующая логика.

1. Подсчет суммы всех одинаковых слов:

```
int sum = 0;
for (IntWritable val : values) {
    sum += val.get();
}
```

2. Если общее количество вхождений слова больше определенного числа, запись этого слова в результат:

```
if (sum > 100)  context.write(key, new IntWritable(sum));
```

Исходный код класса TopWordReducer:

```
package com.sample;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class TopWordReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
                int sum = 0;
                for (IntWritable val : values) {
                        sum += val.get();
                }
                if (sum > 30)       context.write(key, new IntWritable(sum));
    }
}
```

После определения классов-обработчиков запускаем приложение.

Шаг 1. Создаем экземпляр класса org.apache.hadoop.mapreduce.Job.

Класс Job является описанием работы, выполняемой кластером. Он принимает в конструктор конфигурацию кластера (org.apache.hadoop.conf.Configuration) и название выполняемой работы.

```
Job job = new Job(new Configuration(), "TopWordCount");
```

Шаг 2. Указываем объекту Job классы-обработчики, реализующие логику приложения, являющиеся наследниками классов Mapper и Reducer

```
job.setMapperClass(TopWordMapper.class);
job.setReducerClass(TopWordReducer.class);
```

Шаг 3. Указываем каталог с данными для обработки и каталог, в котором будут отображены результаты работы приложения.

```
job.setInputPath(new Path("input"));
job.setOutputPath(new Path("output"));
```

Исходный код класса для запуска приложения:

```
package com.sample;
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.mapreduce.Job;
public class Main {
  public static void main(String[] arguments) throws IOException, InterruptedException, ClassNotFoundException {
        Job job = new Job(new Configuration(), "TopWordCount");
          job.setMapperClass(TopWordMapper.class);
          job.setReducerClass(TopWordReducer.class);
          job.setInputPath(new Path("input"));
          job.setOutputPath(new Path("output"));
  }
}
```

Необходимо заметить, что задачи подобного рода являются комплексными и требуют интеграции с другими сервисами, системами, приложениями и используемыми технологиями. Даже для незначительных изменений, вносимых, например, в обработку данных, необходимо вносить изменения в исходный код, компилировать классы (получать байт-коды) и осуществлять сборку новой структуры приложения. Для обеспечения более гибкого взаимодействия и настройки, а также предоставления дополнительной функциональности существует инструмент Spring Hadoop [8, 9] – фреймворк, который комбинирует возможности Spring Framework с возможностью Apache Hadoop. Эта платформа также поддерживает всесторонний доступ к данным HDFS через такие языки сценариев виртуальной машины Java (JVM), как Groovy, JRuby, Jython и Rhino. Данная технология традиционно позволяет обеспечить слабую связанность компонентов, тем самым делая решение более гибким и модульным.

Например, конфигурирование работы может быть перенесено в конфигурационные файлы Spring Hadoop. Для создания работы необходимо создать тег <hdp:job />, в котором осуществить настройку работы. Атрибуты input-path и output-path будут указывать на папку с исходными данными и папку для результатов работы соответственно. Атрибуты mapper и reducer отражают полный путь к классам TopWordMapper и TopWordReducer. Сформированный тег будет выглядеть следующим образом:

```
<hdp:job id="TopWordCount"
        input-path="/input/"
        output-path="/output/"
        mapper="com.sample.TopWordMapper"
        reducer="com.sample.TopWordReducer"/>
```

Различные дополнительные параметры конфигурирования Apache Hadoop могут быть описаны в специальном теге <hdp:configuration/>. Формат записи: «параметр=значение». Если нет необходимости изменять стандартные параметры, данный тег может отсутствовать в описании.

Для запуска приложения необходимо описать тег <hdp:job-runner />. При указании атрибута run-at-startup=true запуск будет осуществлен при старте приложения. Атрибут job-ref указывает на экземпляр класса с работой, которую необходимо запустить. Сформированный тег будет выглядеть следующим образом:

```
<hdp:job-runner job-ref="TopWordCount" run-at-startup="true"/>
```

Исходный код сформированного конфигурационного файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:hdp="http://www.springframework.org/schema/hadoop"
     xsi:schemaLocation="
          http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
        http://www.springframework.org/schema/hadoop
    http://www.springframework.org/schema/hadoop/spring-hadoop.xsd">
    <hdp:job id="TopWordCount"
            input-path="/input/"
            output-path="/output/"
            jar-by-class="com.sample.Main"
            mapper="com.sample.WordMapper"
            reducer="com.sample.WordReducer"/>
     <hdp:job-runner job-ref="TopWordCount" run-at-startup="true"/>
     </beans>
```

После создания приложения необходимо собрать его в jar-архив. Запуск работы jar-архива осуществляется следующей командой

```
hadoop jar jar-archiv.jar …
```

ЗАКЛЮЧЕНИЕ

Рассмотрены концепция, архитектура, схема работы и принцип взаимодействия компонентов Apache Hadoop, а также способы его практического применения. Продемонстрирован процесс настройки и запуска Apache Hadoop для ОС Windows (Cygwin) и Ubuntu. Приведены способ конфигурирования приложений на основе Java-классов и подход на основе xml-конфигураций с использованием Spring Hadoop. Приведено описание этапов создания приложения, реализующего парадигму MapReduce.

СПИСОК ЛИТЕРАТУРЫ

[1] What is Apache Hadoop? // MapR: [website]. – 2014. – URL: http://www.mapr.com/products/apache-hadoop (дата обращения: 20.01.2014).

[2] Enterprise Hadoop: the ecosystem of projects // Hortonworks: [website]. – [2011–2014]. – URL: http://hortonworks.com/hadoop/ (дата обращения: 20.01.2014).

[3] Hadoop and Big Data // Cloudera: [website]. – 2014. – URL: http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html (дата обращения: 20.01.2014).

[4] Hadoop scales fast on Google Cloud Platform. – 2014. – URL: https://cloud.google.com/solutions/hadoop/?gclid=CKzJ1NjltL4CFSLbcgodY24AXQ (дата обращения: 20.01.2014).

[5] Apache Hadoop: [offic. website]. – 2014. – URL: http://hadoop.apache.org (дата обращения: 20.04.2013).

[6] Java SE Runtime Environment 7 Downloads // Oracle: [website]. – URL: http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html (дата обращения: 20.04.2013).

[7] Cygwin project // Cygwin: [website]. – [2000–2013]. – URL: http://www.cygwin.com (дата обращения: 20.01.2014).

[8] Документация [по Spring Hadoop] // Spring: [website]. – URL: http://spring.io/docs (дата обращения: 20.04.2013).

[9] Spring for Apache Hadoop // Spring: [website]. – 2014. – URL: http://projects.spring.io/spring-hadoop/ (дата обращения: 20.01.2014).

Ботыгин Игорь Александрович, кандидат технических наук, доцент Национального исследовательского Томского политехнического университета. Основное направление научных исследований – распределенные автоматизированные информационно-коммуникационные системы. Имеет более 200 публикаций. E-mail: bia@tpu.ru

Забейворота Артур Владимирович, магистрант Института кибернетики Национального исследовательского Томского политехнического университета. Основное направление научных исследований – мобильные системы. E-mail: ghmulti@gmail.com

Features of deployment, configuration and application of the apache hadoop tools for windows and unix-like operating systems[*]

I.A. BOTYGIN[1], *A.V. ZABEYVOROTA*[2]

[1] *National Research Tomsk Polytechnic University, 30, Lenin Avenue, Tomsk, 634050, Russian Federation, PhD., e-mail: bia@tpu.ru*
[2] *National Research Tomsk Polytechnic University, 30, Lenin Avenue, Tomsk, 634050, Russian Federation, master student, e-mail:* ghmulti@gmail.com

The relevance of the work is driven by the need to process large amounts of heterogeneous data. The aim is to determine how to set up and use freely available tools for designing distributed systems for storage, analysis and processing based on the Apache Software Foundation Apache Hadoop. Research methods include an experimental analysis of the testing and debugging code developed in specialized framework environments. Algorithmic schemes of the infrastructures required for the functioning of the Apache Hadoop, andways to configure Apache Hadoop for Windows (Cygwin) and Ubuntu are described. The cluster management process is implemented through the Apache Hadoop FS shell interpreter console that runs from the operating system. The basic interaction scheme of Apache Hadoop architecture components in distributed processing is researched. A possibility of forming a cluster of computers with an unlimited scale and parallel execution of tasks is proposed. A comparison of ways to configure applications based on Java-classes and an approach based on xml-configurations using Spring Hadoop - framework that combines the capabilities of Spring Framework with the capability of Apache Hadoop is made. The Spring Hadoop platform capability to provide loose coupling of components and to support a comprehensive data access HDFS is shown. The proposed solution is more flexible and modular. A technological scheme of creating an application that implements the Map Reduce paradigm is proposed and tested.

Keywords: Apache Hadoop, Spring Hadoop, Map Reduce, parallel computing, data storage, scalability, Hadoop Distributed File System, Big Data, Apache Hadoop cluster management, Apache Hadoop cluster monitoring, Apache Hadoop cluster deployment

REFERENCES

[1] What is Apache Hadoop? (2014). Available at: http://www.mapr.com/products/apache-hadoop (Accessed 20 January 2014).

[2] Enterprise Hadoop: the ecosystem of projects (2014). Available at: http://hortonworks.com/hadoop/ (Accessed 20 January 2014).

[3] Hadoop and Big Data (2014). Available at: http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html (Accessed 20 January 2014).

[4] Hadoop scales fast on Google Cloud Platform (2014). Available at: https://cloud.google.com/solutions/hadoop/?gclid=CKzJ1NjltL4CFSLbcgodY24AXQ (Accessed 20 January 2014).

[5] Apache Hadoop: [offic. website] (2013). Available at: http://hadoop.apache.org (Accessed 20 April 2013).

[6] Java SE Runtime Environment 7 Downloads (2013). Available at: http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html (Accessed 20 April 2013).

[7] Cygwin project (2013). Available at: http://www.cygwin.com (Accessed 20 January 2014).

[8] Documentation [for the Spring projects] (2013). Available at: http://www.springsource.org/spring-data/hadoop (Accessed 20 April 2013).

[9] Spring for Apache Hadoop (2014). Available at: http://projects.spring.io/spring-hadoop (Accessed 20 January 2014).

[*] *Manuscript received February 12, 2014 .*