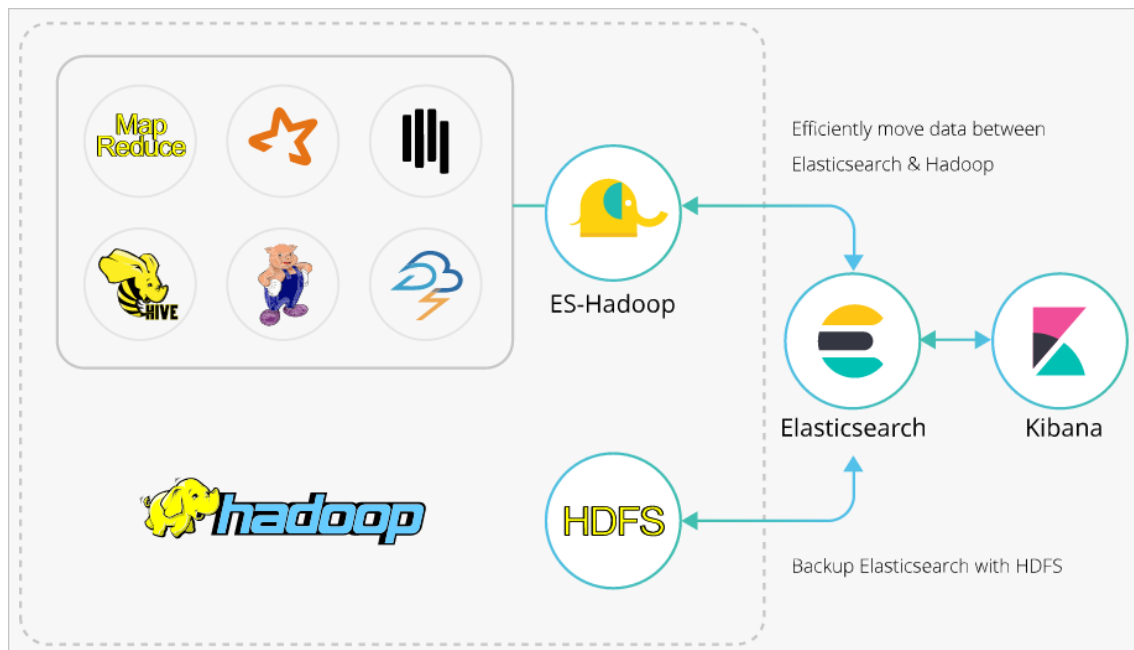


31/12/2022

Big Data Ecosystem Project

Group 10– ES Hadoop MapReduce



Group 10 Members

Benjamin DI SANTO

Aicha BENZEKRI

Antoine CONSO

Florian BETOULLE

Alexandre BIZORT

In this report, we will go over our process to implement a project using Apache Hadoop and Elasticsearch to process large amounts of data.

We will start with a brief presentation of the technologies used, how we implemented the project, what the goal was and finally how the project functions.

1. Technologies used for this project

In this project, we used Apache Hadoop and Elasticsearch Hadoop MapReduce (ES-MR) technologies.

When we need to collect, process/transform, and/or store thousands of gigabytes of data, thousands of terabytes, or even more, Hadoop can be the appropriate tool for the job. With Apache Hadoop, users can do some of the following processes to accelerate their data manipulation:

- Use a cluster of computers so that data can be processed with parallelization and the job can be finished much faster.
- The amount of data generated today keeps growing, so Hadoop is able to easily and flexibly expand its storage capacity too, to keep up with demand. Every computer added to the cluster expands the total storage space available to the Hadoop Distributed File System (HDFS).

To summarize, Hadoop is very good at ingesting and processing incredibly large volumes of data. It distributes data across multiple nodes available in a cluster and uses the MapReduce model to process it on multiple machines at once.

Here is where it is interesting to mention Elasticsearch, as it has tools that can do very similar actions.

“Elasticsearch is a distributed, free and open search and analytics engine, built on Apache Lucene, for all types of data, including textual, numerical, geospatial, structured, and unstructured. It is well known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the central component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization. The Elasticsearch Stack is commonly referred to as the ELK Stack

(after Elasticsearch, Logstash, and Kibana).” – Quoted from the Elasticsearch website.

Of course, if the situation allows it, Hadoop and Elasticsearch can also be teamed up, so we can get the best of both worlds. Hadoop is designed for large collection of data, and can be configured easily to send it to be stored on an Elasticsearch cluster. Elasticsearch would then be great at quickly returning results to the users that search through that data.

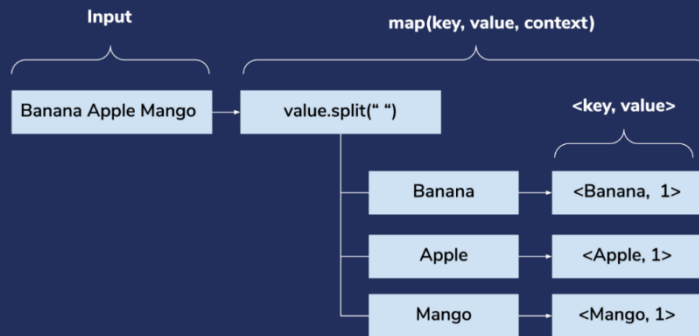
To summarize, with Elasticsearch Hadoop, it is possible to collect and process large amounts of data, in a very efficient manner, and allow for complex, fine-tuned data processing on a clear data visualization tool (Kibana).

How does MapReduce work?

A MapReduce procedure typically consists of three main stages: Map, Shuffle and Reduce.

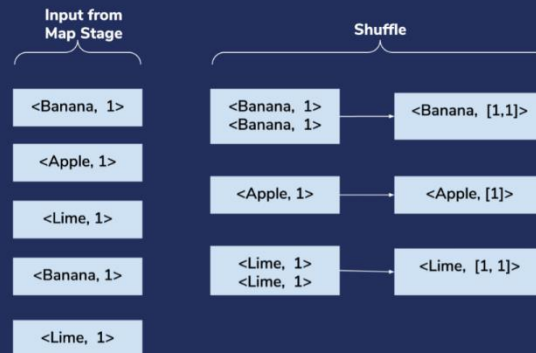
1. Map Stage: Initially, data is split into smaller chunks that can be spread across different computing nodes. Next, every node can execute a map task on its received chunk of data. This kind of parallel processing greatly speeds up the procedure. The more nodes the cluster has, the faster the job can be done.
2. Pieces of mapped data, in the form of key/value pairs, now sit on different servers. All the values with the same key need to be grouped together. This is the shuffle stage.
3. Next, shuffled data goes through the reduce stage, which applies a custom logic to the data and reduced the number of (key, value) pairs.

Map Stage



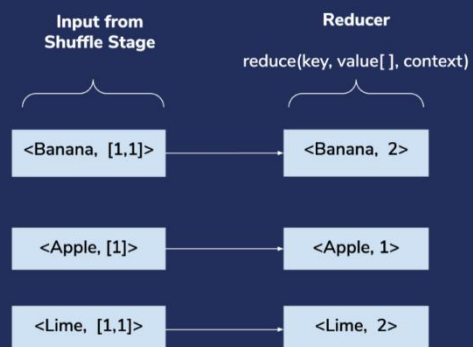
 Coralogix

Shuffle Stage



 Coralogix

Reduce Stage



 Coralogix

2. Implementation of the Project

To implement our project, we used a Linux Ubuntu 22.04 ISO with VirtualBox and the Kibana tool from Elasticsearch on a Windows 10 PC.

First, we needed to initialize and setup the Virtual Machine with the necessary installations including Java, Hadoop and Elasticsearch Hadoop Map-Reduce. Once the necessary dependencies and libraries installed, we could move on to generating our MapReduce project, the essential link between our Hadoop and the Elasticsearch instances. This was a simple Java project that was compiled using Maven and attaching the necessary dependencies to the project artifact. Image below.

```
pom.xml > project
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5      <modelVersion>4.0.0</modelVersion>
6
7      <developers>
8          <developer>
9              <name>Benjamin DI SANTO</name>
10             </developer>
11         </developers>
12
13         <groupId>com.hadoop_elasticsearch_project</groupId>
14         <artifactId>hadoopwithelasticsearch</artifactId>
15         <version>1.0-SNAPSHOT</version>
16
17         <properties>
18             <maven.compiler.source>1.8</maven.compiler.source>
19             <maven.compiler.target>1.8</maven.compiler.target>
20         </properties>
21
22
23         <dependencies>
24             <dependency>
25                 <groupId>org.apache.hadoop</groupId>
26                 <artifactId>hadoop-client</artifactId>
27                 <version>3.3.1</version>
28             </dependency>
29             <dependency>
30                 <groupId>org.elasticsearch</groupId>
31                 <artifactId>elasticsearch-hadoop-mr</artifactId>
32                 <version>8.5.3</version>
33             </dependency>
34             <dependency>
35                 <groupId>commons-httpclient</groupId>
36                 <artifactId>commons-httpclient</artifactId>
37                 <version>3.1</version>
38             </dependency>
39         </dependencies>
40     </project>
```

The detailed code for the MapReduce is in the GitHub repository, with comments.

Once the JAR file of our project was compiled, we moved it to our Hadoop VM instance to be executed. At the same time, we have a Kibana instance that is running to recuperate the output data for visualization.

3. Objective of the Project

The goal of this project is to be able to index a sample access log file which was generated in the Apache Combined Log Format. Unfortunately, a simple log file isn't really a huge amount of data since its only text. However, to properly index every event recorded inside the log, we need to extract every line of text within the file which corresponds to a total of 10000 records.

For example, one log within the file looks like this (the complete file is found in the GitHub repository – “access.log”) :

```
77.0.42.68 - - [17/May/2015:23:05:48 +0000] "GET /favicon.ico HTTP/1.1" 200 3638 "-"
"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:27.0) Gecko/20100101 Firefox/27.0"
```

This is a lot of information within one record, so we need to extract the relevant information for visualization with Kibana, especially the date and time values. To accomplish this, we run the following command for to determine the Elasticsearch index:

```
curl -X PUT "localhost:9200/logs?pretty" -H 'Content-Type: application/json' -d'
```

```
{
  "mappings" : {
    "properties" : {
      "ip" : { "type" : "keyword" },
      "dateTime" : { "type" : "date", "format" : "dd/MMM/yyyy:HH:mm:ss" },
      "httpStatus" : { "type" : "keyword" },
      "url" : { "type" : "keyword" },
      "responseCode" : { "type" : "keyword" },
      "size" : { "type" : "integer" }
    }
  }
}
```

4. Project application with Kibana

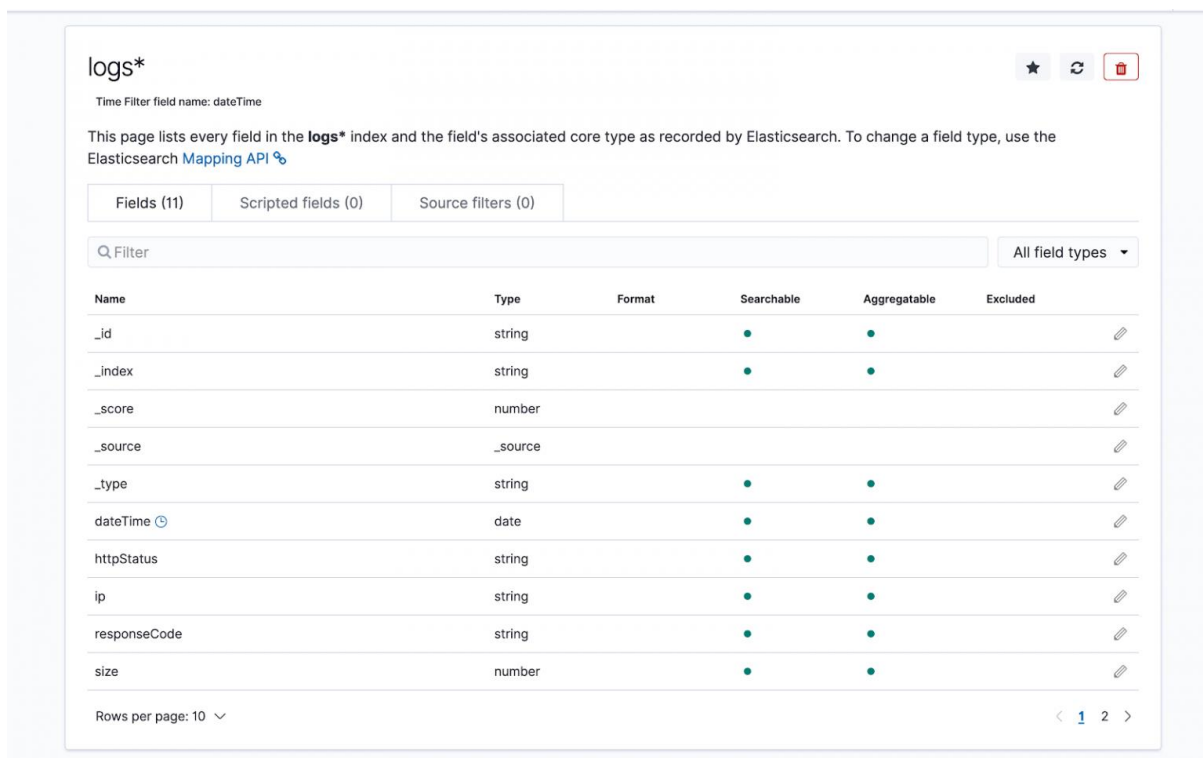
Now that our data is correctly indexed for Elasticsearch and that our JAR file is ready for execution, we can run the following command :

```
hadoop jar hadoopwithelasticsearch-1.0-SNAPSHOT.jar access.log
```

After the command is completed, we can look at the output of the MapReduce job within the file “*joboutput.log*” and assure ourselves the records are correctly indexed with Elasticsearch with the following command:

```
curl 'localhost:9200/_cat/indices?v'
```

Now, to manipulate the data with Kibana, we need to first apply an Index Pattern which filters the data based on date and time, as they are the most interesting parameters to analyze the data with.



The screenshot shows the Kibana Index Pattern page for the pattern `logs*`. It displays a table of fields indexed in Elasticsearch. The table has columns for Name, Type, Format, Searchable, Aggregatable, and Excluded. The 'logs*' index is selected, and the time filter field is set to 'dateTime'. The table lists 11 fields, including system fields like `_id`, `_index`, `_score`, `_source`, and `_type`, as well as application-specific fields like `dateTime`, `httpStatus`, `ip`, `responseCode`, and `size`. All fields are marked as searchable and aggregatable.

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|---------------------------|----------------------|--------|------------|--------------|----------|
| <code>_id</code> | string | | • | • | |
| <code>_index</code> | string | | • | • | |
| <code>_score</code> | number | | | | |
| <code>_source</code> | <code>_source</code> | | | | |
| <code>_type</code> | string | | • | • | |
| <code>dateTime</code> | date | | • | • | |
| <code>httpStatus</code> | string | | • | • | |
| <code>ip</code> | string | | • | • | |
| <code>responseCode</code> | string | | • | • | |
| <code>size</code> | number | | • | • | |

With this done and the assurance that our data was correctly sent it to Kibana, we can analyze our data as we please.

For instance, we can choose to filter the logs which had a size greater than 5MB:

