University of Pisa
School of Engineering

Cloud Computing

A MapReduce implementation of Bloom filters

Supervisors
Nicola Tonellotto
Carlo Vallati

Students
Marco Pinna
Olgerti Xhanej
Federico Cristofani
Matteo Biondi

June 30, 2022

Contents

1 Introduction **2**

2 Overview and design choices **4**
2.1 Bloom filters 4
2.2 Algorithm design 4

3 Hadoop **5**

4 Spark **6**

5 Performance **7**

Chapter 1

Introduction

This work concerns the design, implementation and testing of Bloom filters using the MapReduce framework, both in Java and in Python.
The specifications are detailed in the following:

> You will build a bloom filter over the ratings of movies listed in the IMDb datasets. The average ratings are rounded to the closest integer value, and you will compute a bloom filter for each rating value.
> In your Hadoop implementation, you must use the following classes:
>
> - `org.apache.hadoop.mapreduce.lib.input.NLineInputFormat`: splits N lines of input as one split;
> - `org.apache.hadoop.util.hash.Hash.MURMUR_HASH`: the hash function family to use.
>
> In your Spark implementation, you must use/implement analogous classes.
> In this project you must:
>
> 1. design a MapReduce algorithm (using pseudocode) to implement the bloom filter construction;
> 2. implement the MapReduce bloom filter construction algorithm using the Hadoop framework;
> 3. implement the MapReduce bloom filter construction algorithm using the Spark framework;
> 4. test both implementations on the IMDb ratings dataset, computing the exact number of false positives for each rating;
> 5. write a project report detailing your design, implementations and reporting the experimental results.

The work is organized as follows:

- In chapter 2 a brief overview of Bloom filters is given. Then the general algorithm is presented, with two possible implementations, together with design choices and hypotheses or considerations that have been made about the dataset to be used or the use-cases of the Bloom filters.

- Chapter 3 concerns the Java implementation of the algorithm, that makes use of Hadoop framework.
- Chapter 4 concerns the Python implementation of the algorithm, that makes use of Spark framework
- In chapter 5 an evaluation of the performance of both implementation is provided.

The entire codebase is available at `https://github.com/MPinna/MapReduceBloomFilter`
.

Chapter 2

Overview and design choices

In this chapter we firstly present a brief overview of what Bloom filters are (partly taken from the project specifications); then we show the general algorithm, with two possible implementation along with their respective pseudo-code, together with design choices and hypotheses that have been made during the design process.
Finally, some considerations about the dataset to be used and the use cases of the Bloom filter are made.

2.1 Bloom filters

A *Bloom filter* is a space-efficient probabilistic data structure that is used for membership testing.
A Bloom filter consists of a bit-vector with m elements and k hash functions to map n keys to the m elements of the bit-vector.
The two possible operations that can be done on a Bloom filter are `add` and `test`.
Given a key id_i, every hash function $h_1, ..., h_k$ computes the corresponding output positions and sets the corresponding bit in that position to 1.
The space efficiency of Bloom filters comes at the cost of having a non-zero probability of false positives. The false positive rate of a Bloom filter is denoted by p.
Therefore the two possible outcomes of the `test` function of the Bloom filter are "Possibly in set" or "Definitely not in set".

2.2 Algorithm design

The general idea of this MapReduce implementation is

Chapter 3

Hadoop

Chapter 4

Spark

Chapter 5

Performance