Bloom Filters in MapReduce

A *bloom filter* is a space-efficient probabilistic data structure that is used for membership testing.

To keep it simple, its main usage is to “remember” which keys were given to it. For example you can add the keys “banana”, “apple” and “lemon” to a newly created Bloom Filter. And then later on, if you ask it whether it contains “banana” it should reply true, but if you ask it for “pineapple” it should reply false.

A bloom filter is space-efficient, meaning that it needs very little memory to remember what you gave it. Actually it has a fixed size, so you even get to decide how much memory you want it to use, independently of how many keys you will pass to it later. Of course this comes at a certain cost, which is the possibility of having *false positives*. But there can never be false negatives. This means that in our fruity example, if you ask it for “banana” it has to reply true. But is you ask it for “pineapple”, there is a small chance that it might reply true instead of false.

For more information about bloom filters, check the corresponding Wikipedia page. I will only explain here what are the characteristics and how to use them in a simple calculation to estimate the false positive rate.

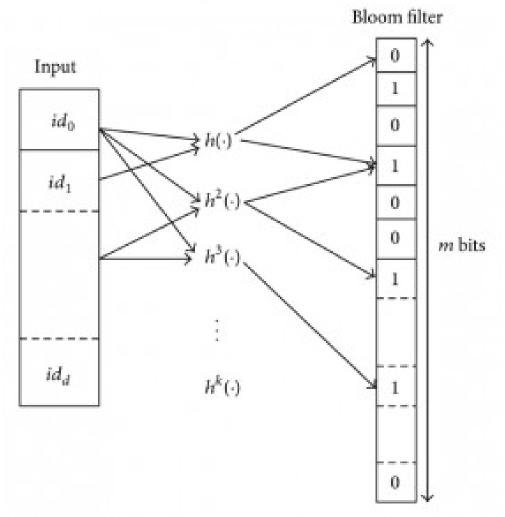

Figure 1: *Bloom filter example.*

A bloom filter is a bit-vector with m elements. It uses k hash functions to map n keys to the m elements of the bit-vector. Given a key id_i, every hash function $h_1, \dots, h_k$ computes the corresponding output positions, and sets the corresponding bit in that position to 1, if it is equal to 0.

Let’s consider a Bloom filter with the following characteristics:

- m : number of bits in the bit-vector,
- k : number of hash functions,
- n : number of keys added for membership testing,
- p : false positive rate (probability between 0 and 1).

The relations between theses values can be expressed as:

$$
\begin{aligned}
m &= -\frac{n \ln p}{(\ln 2)^2} \\
k &= \frac{m}{n} \ln 2 \\
p &\approx \left(1 - e^{-\frac{kn}{m}}\right)^k
\end{aligned}
\tag{1}
$$

To design a bloom filter with a given false positive rate p, you need to estimate the number of keys n to be added to the bloom filter, then compute the number of bits m in the bloom filter and finally compute the number of hash functions k to use.

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.