University of Zürich

Distributed Database

Communication Cost Comparison in 3-Site Semi-Join using Composite Bloom Filters

Andreas Schaufelbühl, Mirko Richter

December 14, 2016

1 Introduction

The basis for this project is a paper that extends the potential of bloom filters from basic two-sites database joins to sites ¿ 2 settings (Michael 2007). Generally, in a distributed database setting bloom filters can be used to reduce the communication costs. Bloom filters can represent a dataset in a compressed way by hashing the data into a bit array, which we call the bloom filter. The result of hashing a data point gives the index or indices in the bit array which corresponds to this data point. This leads to two important properties of bloom filters. First, bloom filters can not be used to send data from one site to another. It rather represents a set of data points such that at a different site one can verify whether a given data point is included in the bloom filter or not. Secondly, when used for verifying whether a given data point is in a bloom filter, the bloom filter can actually give a false positive. Meaning, it claims that this data point is included in the set while it actually is not. As a result, it is not always advisable to use bloom filters. Michael (2007) shows us that there are three variables which contribute to the probability that a bloom filter return a false positive. First, the amount of data to be represented matters as well as the size or length of the bloom filter and the number of hash functions used. The paper already suggests an optimal relation between those variables for a simple bloom filter join between two sites and it goes on an shows how the bloom filters can be combined when there are multiple sites such that one can reduce communication cost even further. This leads us to our project where we set up an experiment to verify some claims made in Michael (2007).

2 Problem Description

A join in a distributed database setting can lead to a lot of communication between the different sites. To reduce the communication cost, one can use bloom filters instead of sending the data points plainly. But since bloom filters do not contain data points but only a representation of them and can return false positives, it is not obvious that a join with bloom filters will perform better.

Michael (2007) provides an optimal relation

$$k \approx \frac{m}{n} * ln(2) \tag{1}$$

between k the number of hash functions, m the size of the bloom filter and n the number of data points. This is for a bloom filter based join over two sites, this implies that there is only one bloom filter to construct. If we now consider a setting in which we want to join over data points distributed over more than 2 sites, Michael (2007) suggests building composed bloom filters. In our project we only consider the intersection[1] of two bloom filters. First of, to be able to intersect two bloom filters, they must be constructed the same way, in other words, they have to have the same length and use the identical hash

[1] Micheal (2007) also covers the union of two bloom filter.

functions.
Then, to build a composed bloom filter bf_c representing the intersection of two bloom filter bf_1 and bf_2 we set a index i in bf_c as follows

$$bf_c[i] = \begin{cases} 1, & bf_1[i] == 1 \wedge bf_2[i] == 1 \\ 0, & otherwise \end{cases} \tag{2}$$

Let us consider a master $site_m$ which wants to join data from two other sites, $site_1$ and $site_2$. Now instead of performing a bloom filter based join with both sites to get the data, the master site can demand the bloom filter from both sites, build the intersection bloom filter and send it back, so that both sites use the composed bloom filter to find join matches. The main focus of our project is this composed bloom filter. Since in the distributed setting the number of data points needed in the composed bloom filter is not available when $site_1$ and $site_2$ are building the bloom filters based on their data, it is not clear how $site_1$ and $site_2$ should choose the number of hash functions and the size of the bloom filter to minimize the communication cost. Michael (2007) provides theory to pre-compute error probabilities for composed bloom filters. For the intersection of two bloom filter, as we have established above, a bit at some index is set to 1 if and only if at the same index both bloom filters is set to 1. If we assume the distribution of our data points to be independent, we can combine the probabilities as follows

$$\begin{aligned} P[bf_c[i] \text{ is set to } 1] &= P[bf_1[i] \text{ is set to } 1 \wedge bf_2[i] \text{ is set to } 1] \\ &= P[bf_1[i] \text{ is set to } 1] * P[bf_2[i] \text{ is set to } 1] \end{aligned} \tag{3}$$

what do we want to show/examine?? that's question and what is paper's answer??

3 Experiment Set Up

4 Implementation

5 Evaluation

6 Conclusion