Hashing and Bloom filters key lookup comparison

names

March 14, 2020

Abstract

1 Introduction

Hashing methods and probabilistic data structures such as Bloom filters have and increasing use nowadays with the necessity of fast and efficient data structures for dictionary abstraction. Dictionaries provide use with a way to organize data and enable a fast search and manipulation. In this report we wish to compare different different hash table implementations and bloom filters in the task of key lookup in the dictionary. In this research we compare three types of dictionaries implementations Open Addressing Tables, Separate Chaining Tables and Bloom Filters.

2 Experiment pipeline and methodology

The experimentation process was a very tedious task since it required to take in account a lot of parameters, ...

Parameters List

Parameter	Definition
n	This is the number of keys that where inserted into the table
m	This is the size of the table
α	The table load factor ($\frac{n}{m}$)
k	Number of hash functions to use for the bloom filters
$keyPercentage$	The percentage of keys that appear in the text files
$seed$	The initial number for the random integer generation

3 Data Generation

The data used in each of the experiments consisted of a sequence of unsigned integers, the data was written in two different files, to distinguish them we used a specific format specified in figure X. One file contained the keys to insert into the dictionary and the other the text to find in the dictionary. The keys file contained n numbers and the text contained $2*n+p*n$ where p is the percentage of keys that are already inserted in the dictionary.

Linear Congruential Method

The randomness of the keys was based on the Linear congruential method (LCM). We needed to have experiments as reliable as possible thus we needed a way to control the data generation, that is why we used this method. The LCM is one of the oldest and best-known random number generators, it is also very easy to implement. This method generates a cyclic sequence of numbers based on an initial seed. Given a seed X_0 it is possible to generate the next number in the sequence

$$X_i = (X_{i-1} * a + c)\, modm \tag{1}$$

where a is called the multiplier, c the incrementer and m the modulus. The size of the sequence depends on the parameters used, independently on the initial seed. The best parameters are the ones that yield the maximum size sequence, that is parameters that yield m numbers. The parameters that we used where ...

4 Experiment parameters

5 Results

6 Discussion