PROJECT PROPOSAL

Apoorv WALIA[1]
Avi SINGHAL[2]

[1]Rice University, Houston, TX

[2]Rice University, Houston, TX
apoorv.walia@rice.edu, as278@rice.edu

Abstract –

1 Background and Motivation

Blocked Bloom filters provide better cache efficiency than standard Bloom filters. However, Blocked Bloom filters have no notion of co-relation between the inputs. They improve cache efficiency by inserting all keys of one input into a single block. Since the size of the block is set to that of the cache line, this improves cache efficiency for insertion and querying of a single key. However, sequential keys could be stored in different blocks and in the worst case, a new block may have to be queried for each new key. This would adversely affect cache performance.

2 Relevance and related works

There are many different implementations of blocked bloom filter and LSH. Elakkiya [3] implemented a cache efficient one hashing blocked bloom filter, the approach included further dividing each of the blocks into partitions of the size of a prime number. They implemented this bloom filter on genomic data and demonstrated better results that prior work. Yu Hua [1] implemented the Locality sensitive bloom filter where they replaced the traditional uniform and independent hash functions with locality sensitive hash functions. They tried to improve the false positive rate by introducing a bit vector to verify the multiple attributes of a member. Lailong [2] provides a detailed survey of the challenges, solutions and also compares various bloom filter implementations.

3 Methods and Approach

Our aim is to improve the performance of Blocked Bloom filters by incorporating Locality-Sensitive Hashing. We intend to pass our input through an LSH algorithm to determine a block for the blocked bloom filter. Once this block is assigned, we will insert the key into the block chosen by the LSH algorithm. The key advantage of this approach is that it leverages the property of LSH algorithms wherein similar inputs will be hashed to the same block. A challenge we may face with this approach is that certain blocks may be overloaded if the input contains a large subsection of similar data. We could tackle this with certain data preprocessing approaches such as normalization or by nesting more blocks inside an overloaded block. As our baseline, we will compare our Bloom filter with a standard Blocked Bloom filter.

We will begin by reviewing the current literature on LSH and Blocked Bloom filters in the first two weeks. We will then implement the current state of the art solutions in the following two weeks. Finally, we will spend our remaining time implementing our idea and comparing our approach with current best solutions.

4 Expected Results and Impact

We expect to build a hashing algorithm that improves upon the cache efficiency of Blocked Bloom filters for datasets that contain similarity between sequential inputs. This is incresingly valauble in applications such as genome sequencing and could vastly improve cache/memory performance of these algorithms.

4.1 Broader Impacts

If the proposed approach succeeds then it can be heavily used for genome sequencing and other applications where data has similarity.

The Broader Impacts criterion encompasses the potential to benefit society and contribute to the achievement of specific, desired societal outcomes. Consider the following questions.

1. What is the potential for the proposed activity to benefit society or advance desired societal outcomes?

2. To what extent do the proposed activities suggest and explore creative, original or potentially transformative concepts?

3. Is the plan for carrying out the proposed activities well-reasoned, well-organized and based on sound rationale? Does

the plan incorporate a mechanism to assess success?

4. How well qualified is the individual, team or institution to conduct the proposed activities?

References

[1] Yu Hua et al. “Locality-Sensitive Bloom Filter for Approximate Membership Query”. In: *IEEE Transactions on Computers* 61.6 (2012), pp. 817–830. DOI: 10.1109/TC.2011.108.

[2] Lailong Luo et al. “Optimizing Bloom Filter: Challenges, Solutions, and Comparisons”. In: *IEEE Communications Surveys Tutorials* 21.2 (2019), pp. 1912–1949. DOI: 10.1109/COMST.2018.2889329.

[3] Elakkiya Prakasam and Arun Manoharan. “A Cache Efficient One Hashing Blocked Bloom Filter (OHBB) for Random Strings and the K-mer Strings in DNA Sequence”. In: *Symmetry* 14.9 (2022). ISSN: 2073-8994. DOI: 10.3390/sym14091911. URL: https://www.mdpi.com/2073-8994/14/9/1911.