Project Specification Document

Title: Improving Co-operative Caching Using Importance Aware Bloom Filter

Advisor Name: Dr. Prof. Hans-Peter Bischof

Student Name: Shridhar Bhalekar (snb3300@rit.edu)

Introduction

Cooperative caching is widely used in distributed environments to reduce the network traffic and increase the query response time. These distributed caching environments use probabilistic data structure like Bloom Filters to store the remote cache information. Hence, the reliability and efficiency of such systems is completely dependent on these Bloom Filters. Bloom Filters have different variants and they can be deployed in different distributed systems.

Bloom Filter

Bloom Filter [4] is a memory efficient probabilistic data structure which might give a false positive result to set (S. Tarkoma, 2012)membership queries. False positives occur when an element is not a member of a given set but a set membership query returns true. Bloom Filter is basically an array of bits representing set elements. Multiple hash functions are used to map items in a set to bits in bloom filter.

Let's say there are n data items to be inserted in the set, there is an array of m bits with all the bits set to 0 and k hash functions. Whenever an element needs to be added in the set, that element is passed to the hash functions to get k values in the range of [0, m). These k values are used as indices and every bit at these k indices is set to 1. To check if any element is present in the set, that element is passed to the k hash functions getting k indices. If all the bits at these positions are set in the array, then the element is present in the set. This technique might result into false positives.

Real World Example

Multiple web proxies sharing caches with each other is one of the relevant examples of cooperative caching. These web proxies share their cache and try to handle each other's cache miss thus reducing the query response time and server load. Such a system of web proxies uses internet cache protocol for collaborative caching. But ICP has an overhead of query broadcasting whenever a cache miss occurs and as the number of proxies increases the communication and CPU processing overhead increases.

Project Goal

There are two objectives for this project. First goal is to compare Importance Aware Bloom Filter [3] with the conventional Bloom Filter. Second goal is to compare Summary cache having Importance Aware Bloom Filter with three different cooperative caching algorithms viz. N-chance, Robinhood and Greedy Forwarding.

Importance Aware Bloom Filter is one of the variant of bloom filter where in apart from k hash functions an importance function is also considered to decide the priority of the set data. So Importance Aware Bloom Filter is an array of m integers instead of m bits and each integer represents the priority level.

In Summary cache [1] each client keeps a cache summary of remote client. This summary is stored as Bloom Filter. Project goal is to use Importance Aware Bloom Filter instead of conventional Bloom Filter for the comparison of different cooperative caching algorithms and checking if Summary cache having Importance Aware Bloom Filter is superior to other algorithms.

Project Execution Strategy

First part of the project is to compare Bloom Filter with Importance Aware Bloom Filter. The metric for the comparison is the amount of false positives encountered while querying the system. Simulation will be used to compare the performance of the two bloom filters. Simulator will create a network of clients and server and will simulate the query forwarding on this system to collect the results for comparison.

Second part of the project is to compare Summary cache with Importance Aware Bloom Filter with N-chance, Robinhood and Greedy Forwarding. Metrics for the comparison of these algorithms are the ticks (time needed to access cache, disk and network) and the cache miss ratio. Trace based simulation will be used where in simulator will simulate the traces against the system and obtain results which will be compared with the results of the actual experiments.

Experiment Design

Hypothesis: Importance Aware Bloom Filter is efficient than Bloom Filter in terms of false positives.

This experiment will be carried over a simulated network of clients and server. There will be multiple clients and single server simulated in the system and each client and server will have their own cache component (i.e. a data structure). Following are the steps which will be executed by the simulator for this experiment:

1. Read the experiment parameters like total clients, client cache size, server cache size, server disk size etc.
2. Simulate two systems, first having clients and server with Bloom Filter and second having clients and server with Importance Aware Bloom Filter (clients and server are objects).

3. Forward n requests to randomly selected clients in the system and will keep track of false positives. Request forwarding will be same for both the systems to maintain consistency.
4. Maintain the experiment results for graph plotting.
5. Repeat experiment for different number of clients and storage sizes.

Hypothesis: Summary Cache with Importance Aware Bloom Filter is efficient than N-chance, Robinhood and Greedy Forwarding.

This experiment will be carried over a simulated network of clients and server. There will be multiple clients and single server simulated in the system and each client and server will have their own cache component (i.e. a data structure). Trace based simulation strategy will be used for this experiment. Following are the steps which will be executed by the simulator for this experiment:

1. Read the experiment parameters like total clients, client cache size, server cache size, server disk size etc.
2. Simulate four different systems each implementing one of the four cooperative caching algorithms.
3. Implement Summary Cache with Importance Aware Bloom Filter.
4. Simulate traces for best to worst cases and traces from the research paper.
5. Randomly forward n requests to random clients in all the four systems, trace every request, keep track of total cache hit, miss and the ticks required to serve each request.
6. Maintain the experiment results for graph plotting.
7. Compare the experiment results with the results of the traces.
8. Repeat experiments for different number of clients and storage sizes.

Project Implementation Steps

1. Design and implement simulator
2. Implement Bloom Filter and Importance Aware Bloom Filter for comparison
3. Analyse the results of the comparison
4. Implement Greedy Forwarding, N-chance and Robinhood
5. Implement Summary Cache with Importance Aware Bloom Filter
6. Compare the cooperative caching algorithms
7. Analyze the results of comparison

Expected Results

1. Importance Aware Bloom Filter should perform better as compared to Bloom Filter in terms of false positives.
2. Summary Cache with Importance Aware Bloom Filter should be superior to N-chance, Robinhood and Greedy Forwarding.

Scientific Contribution

The scope of this project is restricted to comparison of cooperative caching algorithms and Bloom Filter variants. These algorithms have already been proposed, but I will be modifying the Summary Cache algorithm to use Importance Aware Bloom Filter. This modified Summary Cache will

then be compared to Greedy Forwarding, N-chance and Robinhood. My contribution through this project is to analyze the efficiency of this modified Summary Cache over other algorithms.

References

[1] **Li Fan, P. C. (2000). Summary cache: a scalable wide-area web cache sharing protocol.** ***IEEE/ACM*** **(pp. 281-293). IEEE/ACM.**

[2] Ming Zhong, P. L. (2008). Optimizing Data Popularity Concious bloom filters. ***PODC '08 Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing.*** **ACM.**

[3] Puru Kulkarni, R. B. (2013). Importance-aware Bloom Filter for Set Membership Queries in Streaming Data. ***COMSNETS, 2013 Fifth International Conference.*** **COMSNETS.**

[4] S. Tarkoma, C. E. (2012). Theory and Practice of Bloom Filters for Distributed Systems. ***IEEE Communications Surveys and Tutorials. Vol. 14, Number 1.*** **IEEE.**