Performance analysis of a length-aware cuckoo filter in FD.io VPP

Krishna Tippur Gururaj (kxt2163)
Advisor: Prof. Minseok Kwon

OBJECTIVE

The aim is to analyze the performance of IP lookup in a software router when a probabilistic filter is used on top of Cisco’s Vector Packet Processing (VPP) technology. The attempt is to implement a filter that can approximate the membership of an element in a set and can fit into the system cache so as to facilitate quick access.

INTRODUCTION

Packet routing is performed by a router. It is how an IP packet is sent from a source to a destination across a network. It is a crucial component in the flow hence it needs to be fast. A list of all possible IP prefixes at a router is to be looked up to determine the next best hop for the packet to take on its way to its destination. This is done by finding the **longest matching prefix**. The sheer number of possibilities in IPv6 addressing (128 bits in length) has brought forth a need to improve the lookup mechanism and reduce unnecessary memory accesses.

CURRENT SCENARIO

The basic VPP implementation relies on a **naive lookup** of the IP prefix table (a hash table) by checking for all possible prefixes in the address space.

A **cuckoo filter** has been used to improve the performance of the basic implementation by eliminating the absent IP prefixes from the lookup criteria. However, this comes with an expected overhead cost of false positives.

**CUCKOO FILTER**

This space-efficient probabilistic filter stores an element’s fingerprint (a partial hash value) instead of the actual value. The algorithm makes use of two hash functions to verify the presence of a fingerprint. During insertion, this value is inserted in one of the free positions determined by either of the two hash functions. If both are occupied, a previously present value in one of those locations is kicked out and reinserted into a different location, until a free location is found and occupied by the fingerprint.

**LENTH-AWARE CUCKOO FILTER**

This modification of a cuckoo filter works on the principle of using different number of hash functions for previously determined popular (occurring more often) and unpopular (occurring less often) members of a set. There are a few IP prefix lengths which occur significantly more often than all others.

The unpopular IP prefix lengths make up most of the lookups while being the ones which ought to occur less. Hence, the goal is to reduce their false positive rate. This is done by hashing the unpopular-length IP prefixes twice (b) and hashing the popular-length IP prefixes once (a).

The size of the filter is kept small enough to fit into the system cache which ensures extremely low latency in running through the algorithm. The filter determines the possible IP prefixes which might be present in the IP lookup table. The longest of the actually present IP prefixes is then used to route the packet out towards its destination.

**RESULTS**

The Y-axis in the first graph shows the lookup rate (in million packets per second), whereas in the second graph, the Y-axis shows the recorded false positive rates.

LACF clearly performs better than cuckoo filter when integrated with VPP. The false positive rate shows a significant reduction. The lookup rate does not exhibit a major increase, however that is as expected. Main aim of reducing unnecessary memory accesses is achieved.

**CONCLUSION**

LACF exhibits a much lower false positive rate as compared to the conventional cuckoo filter. LACF is able to process more packets per second (faster lookup time) than the conventional cuckoo filter.