Scalable Perfect Hashing Schemes and Applications

Fabiano Botelho
UFMG, Brazil

Abstract:

Minimal perfect hash functions are widely used for memory efficient storage and fast retrieval of items from static sets, such as words in natural languages, reserved words in programming languages or interactive systems, universal resource locations (URLs) in web search engines, or item sets in data mining techniques. It can also be an interesting option to index Large Vocabularies in disk. Mainly if the access distribution to the vocabulary entries follows a Zipf distribution.

A perfect hash function (PHF) maps a key set $S$ of $n$ keys into a hash table of size $m$, where $m$ is greater than $n$, without collisions. A minimal perfect hash function (MPHF) is a PHF with the smallest range, i.e., $m=n$. The minimum amount of space to represent a PHF for a given set $S$ is known to be approximately $1.4427$ bits per key.

The objective of this talk is to present time efficient, near space-optimal and scalable perfect hashing algorithm. We will present: (i) an external perfect hashing (EPH) algorithm that has experimentally proven practicality for sets in the order of billions of keys, (ii) the application of the resulting MPHFs to create contiguous IDs for URLs, and (iii) the application of MPHFs to index Large Vocabularies in disk, which is an ongoing work.

The EPH algorithm we will present has the following properties: (i) evaluation of a PHF or a MPHF requires constant time, (ii) the algorithm is simple to describe and implement, and generate the functions in linear time, (iii) the amount of space needed to represent a PHF with $m > 2n$ is $2.7$ bits per key and a MPHF is $3.7$ bits per key, which is around a factor of $2.6$ from the information theoretical minimum of approximately $1.17$ and $1.4427$ bits per key, respectively. We demonstrate the scalability of the external memory based algorithm by constructing minimum perfect hash functions for a set of $1.024$ billion URLs from the World Wide Web of average length $64$ characters in approximately $62$ minutes, using a commodity PC.