Compact Hash Tables

Tim Gubner
CWI, DA

Peter Boncz
CWI, DA

Hash tables are a crucial data structure for accelerating database operations such as Join and GroupBy. In analytical queries hash tables can grow quite large. As they trade off memory usage against access performance, hash tables tend to be sparsely filled to allow short probing sequences (in the average case).

Additionally their size heavily depends on amount of columns stored inside. For example relational Join operations need to store the whole inner relation and GroupBy operations store the keys and aggregates.

Caused by the mere size, hash tables tend to be located low in the memory hierarchy (main memory, disk). We believe that reducing the size of each row will speed up hash table access by allowing to fit more useful data in cache, as well as reduce memory usage.

In our talk we will focus on reducing the per-row overhead by suppressing unused bits in the prefix of each value based on its domain. Additionally we will present two more elaborate techniques for shrinking size of hash table's active working set.

(1) We are going to introduce *Speculative Packing* which splits the hash table into two areas. One area will be frequently used during query execution and is optimized for the average case. The other area will store rarely used values.

(2) These techniques only work effectively for integers and mainly exclude strings. For strings Speculative Packing can be extended using *Smart on-the-fly Dictionaries* which store often used strings and provide indices to unique strings. These indices are stored in the regularly accesses area of the hash table, whereas strings that are not in the dictionary are stored in the rarely accessed area.

Note that this is ongoing work and this talk will concentrate on the underlying ideas. However the reduction of memory usage in the TPC-H benchmark seems promising.