PH-Tree

A Space-Efficient Storage Structure and Multi-Dimensional Index

presented by Kevin Zuniga

Saint Agustin National University

kevin.zun@gmail.com

November 24, 2014

Overview

Introduction

- In this presentation I present the PATRICIA-hypercube-tree.
- It combines binary PATRICIA-tries with a multi-dimensional approach similar to quadtrees while being navigable through hypercubes.

Related Structures

- Quadtrees
- PATRICIA-tries
- Crit(ical)-bit-trees

The PH-Tree

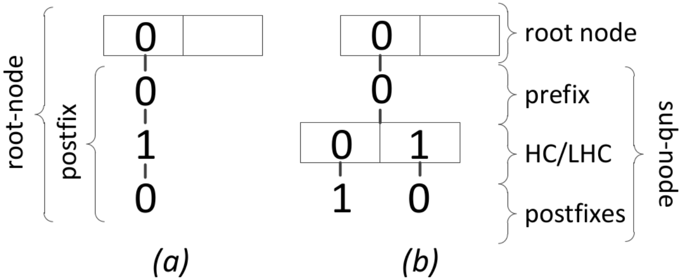
- It is essentially a quadtree that uses hypercubes, prefix-sharing and bit-stream storage.
- Each node can contain up to 2^k children.
- The depth of the tree is independent of k and equal to the number of bits in the longest stored value, w.
- It's unbalanced, it avoids problem with degeneration.
- It does not aim for maximum node occupancy, but reduces the size overhead of nodes.

The 1D-PH-Tree

- A PH-Tree stores *entries*, which are set of *values*.
- The first bit of any value in the tree is stored in the root node.
- There is an array for fast look-up of references to entries and sub-nodes.
- Entries that are attached to an array field without further sub-nodes are called a *postfix*.

1D-PH-Tree

A sample 1D-PH-tree with one 4-bit entry (a) and two 4-bit entries (b).

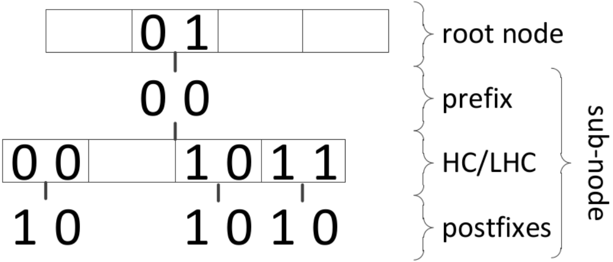


The kD-PH-Tree

- The bit-strings of the values of one entry are stored in parallel.
- The size of the HC is 2^k for a k-dimensional tree.
- We can reduce the memory requirements for high values of k by creating a linear representation of HC, LHC.

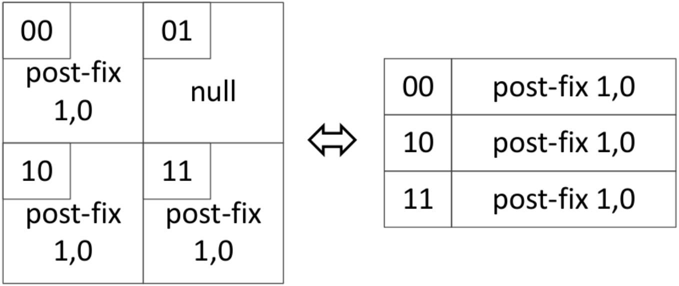
kd-PH-Tree

A sample 2D-PH-tree with three 4-bit entries: (0001, 1000), (0011, 1000), (0011, 1010).



HC and LHC

HC and LHC representation of references in a node.



Floating Point Values

PH-tree works with bit-strings, that sort like integers, therefore floating point numbers must be converted first.

```
long c(double value) {
    long raw = Double.doubleToRawLongBits(value);
    if (value < 0.0) {
      return raw ^ 0x7FFFFFFFFFFFFFFFL;
    }
    return raw;
}
```

The conversion function has the property that for $i_1 = c(f_1)$ and $i_2 = c(f_2)$, $i_1 > i_2 \iff f_1 > f_2$.

Query Efficiency

- The PH-tree supports two types of queries
  - Point queries
  - Range queries
- Query efficiency depends as much on the type of query as on the characteristics of the stored data.

Point Query

- A point query takes an entry as a parameter, and it checks whether an equivalent entry already exist or not.
- We just have to traverse the trie, it takes $O(w * k)$.

Range Query

- A range query takes a query-rectangle defined by a lower left point and an upper right point, it returns and iterator over all points in the rectangle.
- The query starts with a with a point query that locates the starting node.
- Then for each candidate node, all postfixes and subnodes that potentially intersects with the query need to be traversed.

Range Query Worst Case

- The worst case occurs when a query restricts only one or few dimensions out of k and if the values in these dimensions share long prefixes.
- For example a query on a dimension whose only values are 00000000 and 00000001.
- Another worst case occurs when many entries are postfixes of the same node.

Range Query Best Case

- In the best case, location of the staring node is followed by a series of matches until the upper range of the query is reached.
- It results in a time complexity of $O(w * k * n_{matches})$.

Range Query Average Case

- The average query complexity cannot be established as a simple function of n.
- However experiments show that the query complexity tends to vary between $O(\log n)$ and $O(1)$.

Conclusions

- I have presented the PH-Tree as an approach for combined storage and indexing of multi-dimensional data.
- The tests showed that the PH-Tree is very space efficient.
- As a multi-dimensional index structure, it showed competitive performance for queries.
- In summary the combination of multi-dimensional indexing with space efficiency and good performance makes it a useful alternative for many applications.

References

Zäschke, Tilmann and Zimmerli, Christoph and Norrie, Moira C. (2014)
The PH-tree: A Space-efficient Storage Structure and Multi-dimensional Index
Proceedings of the 2014 *ACM SIGMOD International Conference on Management of Data.*

PH-Tree

A Space-Efficient Storage Structure and Multi-Dimensional Index

presented by Kevin Zuniga

Saint Agustin National University

kevin.zun@gmail.com

November 24, 2014