Wormhole: A Fast Ordered Index for In-memory Data Management (II)

Main Paper: Wormhole: A Fast Ordered Index for In-memory Data Management

Authors: Wu, Xingbo, Fan Ni, and Song Jiang.

Published in: In Proceedings of the Fourteenth EuroSys Conference 2019

Published Year: 2019

Publisher: ACM

Presented By,
Jaison , Niya [1001562701]
Overview

• Anchor- A borderline between different leaf nodes in the LeafList.

• Target node for K is a leaf node whose anchor key K1 and immediately next anchor key K2 satisfy K1 ≤ K < K2

• In MetaTrie, to reach the target leaf node of the search key:-
  1. Conduct the longest prefix match (LPM) between the search key and the anchors in the trie.
  2. If the longest prefix is not equal to an anchor :
     a) Traverse on a subtree rooted at a sibling of the token next to the matched prefix of the search key.
• The cost of each of the phases is $O(L)$, which is to be reduced to $O(\log L)$.

• In MetaTrieHT the trie structure is stored as a hash table and all prefixes of anchors in the trie structure is inserted to the hash table.

• In MetaTrieHT the longest prefix match is done using binary search algorithm instead of token by token matching.

• This reduces the search cost to $O(\log L)$.

• The hash table has indices to each of the leaf, which makes look up cost to $O(1)$. 
Q.1) Use Figure 5 as an example to explain how a MetaTrieHT is built.

In Figure 5, the MetaTrie mentioned in the Figure 3 is converted to MetaTrieHT.
• To obtain the LPM, we use a binary search on prefix lengths, for which we insert all prefixes of each anchor into a hash table.

• If the node represents an:
  • Anchor, or a leaf node:
    • the hash item is a leaf item .
    • Represented as ‘L’ .
  • Internal node:
    • the hash item is an internal item
    • Represented as ‘I’ .

• Each hash item has two fields supporting efficient walk in the second search phase on a path to a leaf node.
  • The first field is a bitmap.
  • The second field - two pointers, each pointing to one of the leaf nodes.
Q.2) Assume the anchor “Alexander” is in the MetaTrieHT, and the search key is “Alexandria”. Show how many steps (hash table lookups) are required to complete the first phase.

- To obtain the LPM the authors used an algorithm which performs binary search on prefix lengths.

**Algorithm 1 Binary Search on Prefix Lengths**

1: function searchLPM(search_key, L_anc, L_key)
2:     m ← 0; n ← min(L_anc, L_key)+1
3:     while (m+1) < n do
4:         prefix_len ← (m+n)/2
5:         if search_key[0 : prefix_len-1] is in the trie then
6:             m ← prefix_len
7:         else n ← prefix_len
8:     return search_key[0 : m-1]
Figure: Steps in phase one for the search key “Alexandria”
• Anchor: “Alexander”
• Search_Key = “Alexandria”
• L_{anc} = the length of the anchor = 9
• L_{key} = the length of the search key = 10
• Search_key(‘Alexandria’, 9, 10): m=0 and n=10

<table>
<thead>
<tr>
<th>While(m+1)&lt;n : 0&lt;10</th>
<th>While(5+1)&lt;10</th>
<th>While(7+1)&lt;10</th>
<th>While(7+1)&lt;8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix_len = 5</td>
<td>Prefix_len = 7</td>
<td>Prefix_len = 8</td>
<td>return('Alexand')</td>
</tr>
<tr>
<td>if('Alexa' is in trie): m=5;</td>
<td>if('Alexand' is in trie): m=7;</td>
<td>if('Alexandr' is in trie): else: n=8;</td>
<td></td>
</tr>
</tbody>
</table>
Q.3) Use Figure 5 as an example to show how the search key “Jacob” is found in the tree.

Figure 5: The structure of Wormhole. For clarity the bitmap is depicted by directly listing child tokens.
1. The MetaTriHT will perform LPM using the binary search.
   
   1. Hash("Jacob") would return an index which points to ‘Ja’.

2. But ‘Ja’ is not an anchor node. This means we need to go for the second phase.

3. ‘Ja’ internal node has only one child node, ”Jam”. ‘c’ should have occurred before ‘m’, i.e, ‘m’ is the right sibling tree of ‘c’.

4. So we consider the ‘Jam’ entry in the hash table. As this is an anchor node, we will get the respective leaf node as the target node.
References:


Questions ??
Thank You