ACM SIGMOD Programming Contest 2017

Team: DVA

Hyeonseok Oh, Jongbin Kim, Hyungsoo Jung(Advisor)

{hyeonseok.ohh, mrbin20022, hyungsoo.jung}@gmail.com

1. Task Overview

- The task is to search documents and return strings(N-gram) from a given set, as quickly as possible.
- Firstly, an initial set of N-grams is given. The time spent on processing this initial set is not counted into the total execution time.
- Next, the workload comes in batches. Each batch consists of three different operations.
    1. A <string> : add a N-gram into the list of N-grams
    2. D <string> : delete a N-gram from the list of N-grams
    3. Q <document> : query all N-grams in the document which is in the up-to-date list of N-grams. These should be presented in order of their first appearance in the document.

2. Solution Overview

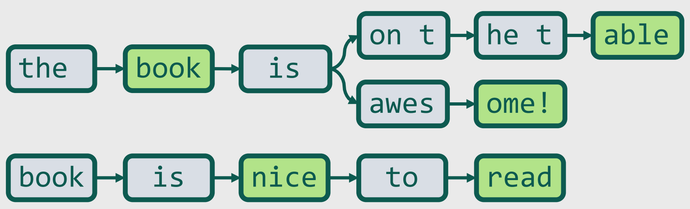
- We improved the performance from following aspects:
    1. Remove redundant searching for a “Query” operation by using a trie structure to represent N-gram data.
    2. Distribute the update workloads to dedicated worker threads: Each Add/Delete worker manipulates a partitioned N-gram trie to run update operations concurrently without having malicious data race.
    3. Distribute the “Query” workloads: Split a document into multiple segments, each of which is assigned to a query worker thread.
    4. Exploit features of N-gram: Reduce memory reference time when traversing a trie by embedding a single edge directly in a node.

3. N-gram Trie

- We maintain a trie to express the N-gram words. Each node of a trie describes 4 characters (4 bytes) so that we can compare 4 characters at a single trie iteration by the Query worker.

N-grams

- the book
- the book is on the table
- the book is awesome!
- book is nice
- book is nice to read



- By using a trie, we remove redundant word comparisons. Just looking forward once to the tries’ lower level is enough.
- We partitioned N-grams by hashing the first character of words to distribute Add/Delete workloads to multiple threads. Each partitioned trie is manipulated by a single thread to avoid the race condition.
- An edge list is managed by a hash table for each node. While maintaining a single version of an N-gram trie, we don’t need to care about concurrent updating workers in the hash table.

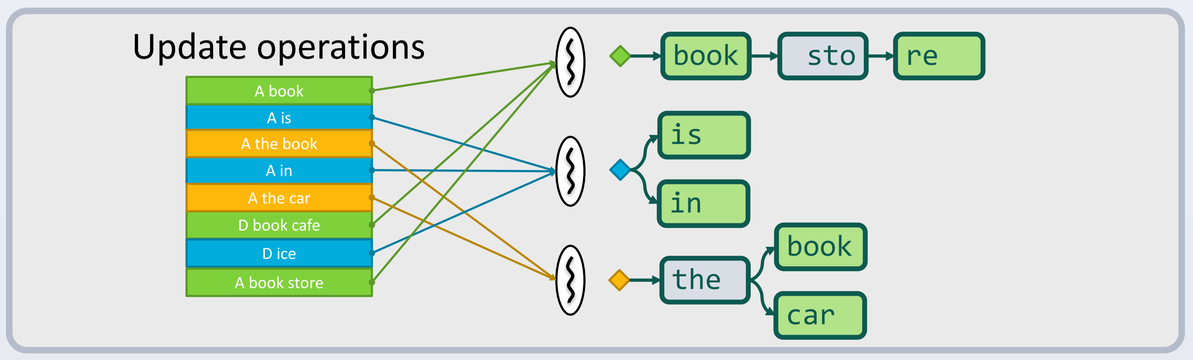
3. Algorithms of Update Operations

Master Thread

1. Choose the thread responsible for the task by hashing the first character of a N-gram.
2. Append the task to the backlog queue of the worker thread.
3. Get the next input and repeat.

Worker Thread

1. Wait until there is a task in their backlog queue.
2. Execute the task using their trie.



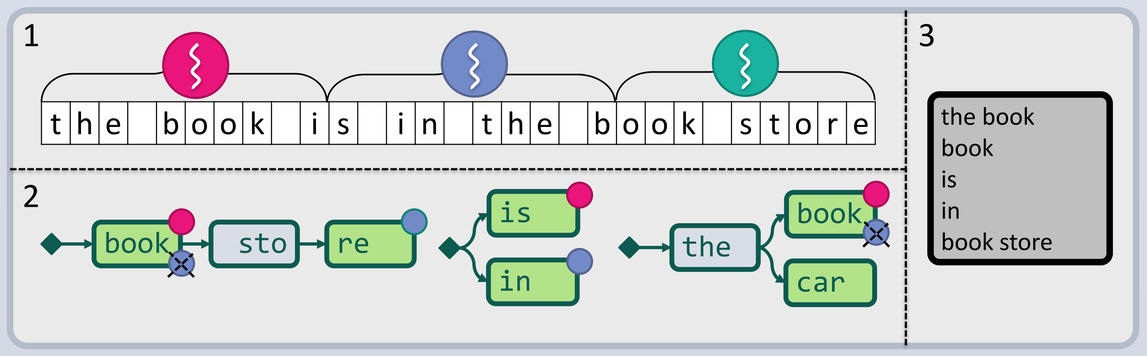
4. Algorithms of Query Operations

Master Thread

1. Broadcast the query to workers by increasing a query counter.
2. Check each worker thread’s result list ordered by the thread ID. If the worker thread has not finished, wait for it.
3. Check the signature in the node to ensure that the signature is the same as the signature of the worker thread.
4. Print the N-gram only If the signature is the same.

Worker Thread

1. Finish the assigned update tasks and set a completion flag.
2. Calculate the range of the document which it has to handle.
3. Find N-grams in a given range. If another worker thread dedicated to handle the N-gram candidate is still processing an update operation, wait until the completion flag is on.
4. Compare the signature in the node and update the signature by CAS if the node is valid.
5. If CAS is succeeded, append the information in the result list. If CAS failed, compare the signature again.



5. Exploit the Features of N-grams

- Probably there are many nodes with only one edge at the lower level of N-gram tree. We keep one out-going edge directly in each node to reduce the memory referencing time.