CS6023: GPU Programming

Assignment 4 (13 marks)

Submission deadline: April 18, 2021, 23:55 on Moodle

1 Problem Statement

We are given a database as input. Write an efficient and data-race free GPU code to perform a concurrent search, range query, addition, and path tracing operations on the database using the B+ tree data-structure on GPU. In other words, create a B+ tree data structure to store the key attribute of the database and process the search, range query, addition, and path tracing operations on GPU concurrently.

Points to be noted:

- The first attribute of the database is always the key attribute for this assignment.
- B+ tree should be created using the key attribute only.
- Fix the fanout of the B+ tree to 8 for this assignment.
- Use the right replication policy while creating the B+ tree.
- We are interested in doing a bunch of each individual operation concurrently on GPU. Different operations may happen in a sequential manner.

The four operations that we are supposed to perform on the database are summarized as follows:

1. Search: Given a key(k), print the tuple corresponding to the given key(k) in the database. If no such key is found, then print -1.
2. Range query: Given a range [a,b], print all the tuples with keys with-in the range [a,b], including a and b. If no keys are present with-in the given range, then print -1.
3. Addition: Given a key(k), an attribute number(anum) and value(v), add the value(v) to the attribute number(anum) of the tuple with key(k). The addition operation does not print anything, but it will make changes to the database.
4. Path tracing: Given a key(k), print the first key value of all the traversed nodes while searching the node with key(k) in the B+ tree.

2 Input format with example

2.1 Example input

21 6
13 0 0 0 0 0
14 0 0 0 0 0
15 0 0 0 0 0
17 0 0 0 0 0
5 0 0 0 0 0
6 0 0 0 0 0
7 0 0 0 0 0
29 0 0 0 0 0
32 0 0 0 0 0

```
33 0 0 0 0 0
9 0 0 0 0 0
10 0 0 0 0 0
11 0 0 0 0 0
18 0 0 0 0 0
20 0 0 0 0 0
21 0 0 0 0 0
25 0 0 0 0 0
28 0 0 0 0 0
1 0 0 0 0 0
2 0 0 0 0 0
4 0 0 0 0 0
4
4 2
2 2 32 80 100 500
3 3 21 4 987 18 3 143 6 2 100
1 3 21 18 6
```

2.2 Explanation

The first line contains two space-separated integers denoting the number of rows(n) and columns(m) in the input database. The following 'n' lines denote each row of the database. The following line contains a single integer 'q' denoting the number of the bunches of queries. In each of the following 'q' lines, the first integer(t) represents the operation type (1:search, 2:range query, 3:addition, 4:path tracing). If the operation is a path tracing operation, then the second integer(k) denotes the key for which we are supposed to trace the path while searching it. If the operation is a search, range query, or addition operation, then the second integer(p) denotes the number of queries of the previously specified operation. The following 'p' integers denote the keys(k) to be searched if it is a search operation. The following '2*p' integers denote the ranges [a,b] if it is a range query operation. The following' 3*p' integers denote the key(k), attribute number(anum), and value(v) to be added if it is an addition operation.

3 Constraints

- $1 \leq n \leq 10^6$
- $1 \leq m \leq 20$
- $1 \leq database[i][j] \leq 10^9$, where $1 \leq i \leq n$ and $1 \leq j \leq m$
- $1 \leq q \leq 1000$
- $t \in \{1,2,3,4\}$
- $1 \leq p \leq 10^4$
- $1 \leq k, a, b \leq 10^9$
- $2 \leq anum \leq m$
- $1 \leq v \leq 1000$
- All other attributes except the key attribute are initialized to zero in the input database.

4 Output Format

- Each query of the search operation prints the single line of 'm' space-separated integers denoting the tuple corresponding to the key that is to be searched. If the required key is not present in the database, then print -1.

- Each query of the range query operation prints the 'z' lines of 'm' space-separated integers denoting the tuples that has its key value within the specified range. Note, 'z' denotes the total number of keys that are within the specified range. If no keys exist with the specified range, then print -1.
- The addition operation does not print anything but will update the database.
- Each query of the path tracing operation prints the single line of 'v' space-separated integers denoting the first key value of all the traversed nodes while searching the node with specified key. Note, 'v' denotes the total number of nodes traversed while searching the given key in the B+ tree.

5 Sample Test-case with Explanation

5.1 Sample Input

```
21 6
13 0 0 0 0 0
14 0 0 0 0 0
15 0 0 0 0 0
17 0 0 0 0 0
5 0 0 0 0 0
6 0 0 0 0 0
7 0 0 0 0 0
29 0 0 0 0 0
32 0 0 0 0 0
33 0 0 0 0 0
9 0 0 0 0 0
10 0 0 0 0 0
11 0 0 0 0 0
18 0 0 0 0 0
20 0 0 0 0 0
21 0 0 0 0 0
25 0 0 0 0 0
28 0 0 0 0 0
1 0 0 0 0 0
2 0 0 0 0 0
4 0 0 0 0 0
4
4 2
2 2 32 80 100 500
3 3 21 4 987 18 3 143 6 2 100
1 3 21 18 6
```

5.2 Sample Output

```
9 1
32 0 0 0 0 0
33 0 0 0 0 0
-1
21 0 0 987 0 0
18 0 143 0 0 0
6 100 0 0 0 0
```

5.3 Explanation

The following figure shows the B+ tree generated by inserting the keys from the sample input database.

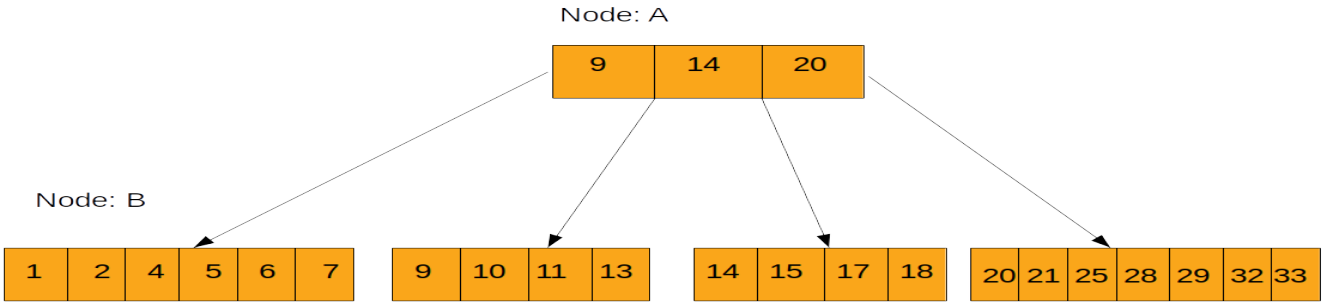

Figure 1: Constructed B+ Tree from the input database

The figure 1 shows the B+ tree constructed from the input database. The first operation that we perform is the path tracing operation with key as 2. To reach the node with the given key, we start from node A and then move to node B, where the required key to be searched is present. The nodes A, B are visited during this operation, and the first key values of those two nodes are 9 and 1, respectively. The second operation performs a range query for the input ranges [32, 80] and [100, 500]. We find two keys present in the first range; hence we print the tuples corresponding to both of them. No keys are present corresponding to the second range; hence we print -1. The third operation adds the value 987 to the fourth attribute of the tuple where key 21 is present. Similarly, it adds the values 143 and 100 to the third and second attributes where keys 18 and 6 are present, respectively. This operation does not print anything. The fourth operation searches for the keys 21, 18, 6. Since all the searched keys are present, it prints the tuples corresponding to those keys. Note how the printed tuples reflect the changes that are done by the previous addition operation.

6 Additional points to be noted

- Test your code on a large input database.
- Those submissions that do not use the B+ tree to perform all the above-specified operations will be discarded and given 0 points.
- Those submissions that perform the operations mentioned above sequentially will also be discarded and given 0 points.

7 Submission guidelines

- Compress the file 'main.cu', which contains the implementation of the above-described functionality to ROLL_NUMBER.tar.gz
- Submit only the ROLL_NUMBER.tar.gz on moodle.
- The name of the file should strictly be of the format ROLL_NUMBER.tar.gz
- For example, if your roll number is CS16D019, the name of the file you submit should be CS16D019.tar.gz
- Make sure that the ROLL_NUMBER part of the filename is in upper case.
- Do not upload anything other than the ROLL_NUMBER.tar.gz file.
- After submission, download the file and make sure it was the one you intended to submit.

8 Learning suggestions

- Write a CPU-version of code achieving the same functionality. Time the CPU code and GPU code separately for large databases and compare the performances.
- Try to develop a GPU-based B+ tree data-structure such that it exploits the memory coalescing on GPU.
- Try reducing thread divergence as much as possible while performing the above-specified operations.
- Exploit shared memory as much as possible to gain performance benefits.
- Try doing different operations concurrently on GPU while preserving the correctness. For example, perform search operation along with the addition operations concurrently such that correctness is preserved.

9 Additional Resources

Below are a few links that might help understand the search and insertion operation in the B+ tree in detail.

- `https://www.cs.helsinki.fi/u/mluukkai/tirak2010/B-tree.pdf`
- `https://www.cs.nmsu.edu/~hcao/teaching/cs582/note/DB2_4_BplusTreeExample.pdf`

The following link might help in visualizing the search and insertion operations in the B+ tree.

- `https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html`