Lab 2 Cuckoo Hashing

Ziyang Wang
ShanghaiTech University
wangzy4@shanghaitech.edu.cn

1. Cuckoo Hashing

Cuckoo Hashing is a hashing method which gurantees O(1) time lookup operations, which could be easily accelerated by GPU.

In this project, I've implemented a Cuckoo hash table in CUDA for GPU accelerating.

The basic algorithm of Cuckoo hashing is introduced in Cuckoo hashing - Wikiwand, which is named by the bird which pushed the other eggs or young out of the nest when it hatches other species' nest.The basic principle of Cuckoo hashing is to apply different hash function on the older key in the hash table and put it into other location when two keys have same hash value with their hash functions (could be same or different).

2. Design Details

Different from the hash table mentioned in the web page, I wrote one hash table for all hash function, and makes the size of table large.

To prevent failed insertion, I implemented a second table(which is much smaller than the original hash table), and use different hash function. Second table is used when evict time is equal to bound. Lookup in the second table is sequential, but because of the size of table is known(and could be restricted when inserting), the lookup time is still $O(1)$.After implementing the second table, failed insertion would be much less than just a single table.

For the blocksize and gridsize, I use $cudaOccupancyMaxPotentialBlockSize$ in CUDA to decide, to make sure the occupancy will be as large as possible.

3. Computer Configuration

I use the AI cluster of the university, the GPU information is:

Name	Tesla K80
Compute capability	3.7
Clock rate	823500
Device copy overlap	Enabled
Kernel execition timeout	Disabled
Total global mem	11996954624
Total constant Mem	65536
Max mem pitch	2147483647
Texture Alignment	512
Multiprocessor count	13
Shared mem per mp	49152
Registers per mp	65536
Threads in warp	32
Max threads per block	1024
Max thread dimensions	(1024, 1024, 64)
Max grid dimensions	(2147483647, 65535, 65535)

4. Results

I use $cudaEvent$ for most of the testcases, but it will cause segmentation falut when I add $cudaEventSynchronize(stop)$ in testcase 2. After deleting this line, the program will but run normally but with the wrong time.

Therefore, I use function in "$time.h$" with $cudaDeviceSynchronize()$ for testcase 2.

Here's the result of four testcases, the original result and code is in the directory with the number of corresponding testcase.

1.

Hash Function	s	Time/ms
2	10	4.99
2	11	4.39
2	12	3.33
2	13	3.22
2	14	4.70
2	15	3.31
2	16	4.75
2	17	4.27
2	18	4.61
2	19	3.04
2	20	7.21
2	21	6.95
2	22	18.63
2	23	39.09
2	24	107.67
3	10	1.13
3	11	1.17
3	12	1.22
3	13	1.26
3	14	1.21
3	15	1.15
3	16	1.59
3	17	1.31
3	18	1.69
3	19	1.79
3	20	3.02
3	21	5.39
3	22	9.62
3	23	18.77
3	24	37.65

Obviously, using 3 hash function will be mush faster than two hash functions, due to the larger hit(value is empty) rate, while increasing number of keys will not cause slower runtime when number of key is less than 2^{20}.

2.

Hash Functions	i	Time/ms	Hash Functions	Time/μs
2	0	10	3	5
2	1	7	3	5
2	2	7	3	8
2	3	6	3	6
2	4	6	3	22
2	5	6	3	7
2	6	6	3	5
2	7	7	3	7
2	8	6	3	5
2	9	6	3	7
2	10	7	3	7

Because lookup time is always in $O(1)$ time, there aren't much difference between different keys.

3.

Because the size of table is too close to key size, it always failed when there's only two hash functions.

Tablesize	Time/ms
1.1	44.81
1.2	41.07
1.3	37.81
1.4	35.98
1.5	35.57
1.6	34.29
1.7	33.90
1.8	33.01
1.9	33.09
2	32.69
1.01	51.57
1.02	49.42
1.03	49.07
1.04	48.81
1.05	47.57

Insertion time is decreased with the size of table.

4.

hash functions	bound/(log n)	time/ms	hash functions	time/ms
3	1	33.36	2	26.19
3	2	33.07	2	33.46
3	3	33.43	2	48.23
3	4	33.86	2	55.40
3	5	3.85	2	81.56
3	6	34.03	2	92.55
3	7	34.77	2	101.20
3	8	34.58	2	112.08
3	9	35.14	2	123.14
3	10	35.54	2	132.13

Time cost doesn't change a lot with 3 hash function, but increases with the bound when there're only 2 has functions, which might because of the probability of table with two hash function to put key into second table is much higher than 3 hash function tables.

5. Reference

1.Cuckoo hashing - Wikiwand
2.CUDA timing function details