CS121@Fall2021 Lab 2
Cuckoo Hashing on GPU using CUDA

Cheng Peng (彭程)* 2020533068

December 29, 2021

Contents

*pengcheng2@shanghaitech.edu.cn

1 Introduction

Hash table is one of the most widely used data structure. Since it was invented, extensive studies and practical uses have been made. Separate chaining and open addressing.

Our overall goal is to make the algorithm as fast as possible.

2 The Cuckoo Hashing

3 Algorithm Design

3.1 Choice of Hash Function

To achieve good performance, we have to choose a set of hash function with care. Hash functions of high quality are generally slow to compute, which may limit the throughput. However, using fast hash functions may result in more collisions, which in turns lower our throughput. Finding A hash function that achieves best balance between speed and quality is the key to high performance.

A research[JO20] published on JCGT[1] tested a set of cryptographic and non-cryptographic hash functions for their quality and speed in the context of GPU rendering. They concluded that for 1D to 1D hash function, which is the scenario in lab2, the *xxhash32* falls in the middle of the spectrum from the fastest algorithms to the algorithms that yield the best results.

A simple reference implementation[Bru] in cpp was found.

The *xxhash32* takes a two input a 4-byte seed and a byte stream, then produces a 32-bit integer as output. Or formally speaking:

$$H : \text{byte}^4 \times \text{byte}^* \to \text{byte}^4$$

Therefore, we can creat a set of hash functions by seeding *xxhash32* with different seeds.

$$H_i : \text{byte}^* \to \text{byte}^4 \qquad H_i(\text{text}) = H(\text{seed}_i, \text{text})$$

[1] Journal of Computer Graphics

4 Performance Evaluation

5 Futher Improvement

A Fun Facts

- According to https://developer.nvidia.com/blog/inside-pascal/, atomic operations are optimized for global memory not the shared memory since Kepler SM architecture.
  However, we get neither a boost nor a degrade in performance when testing our program on RTX 2080 Ti with Turing architecture SM.

bibliography

[JO20] Mark Jarzynski and Marc Olano. “Hash Functions for GPU Rendering”. In: *UMBC Computer Science and Electrical Engineering Department* (2020).

[Bru] Stephan Brumme. *xxHash - a hash algorithm as fast as memcpy*. URL: https://create.stephan-brumme.com/xxhash/.