实验报告

实验目的

根据论文《A Fast and Cost-Efficient Hashing Index Scheme for Cloud Storage Systems》提出的一种高性能的哈希索引结构，以及论文中给出的相关开源代码和使用的数据集，重现实验，验证这种哈希结构相对于传统哈希的改进和性能上的提升，从而对该论文有更好的理解。

实验环境

实验开展的环境是：
Ubuntu 16.04 LTS
Intel i5 7200U 2.5GHz 4-core CPU
16GB DDR4 RAM
网卡带宽：1GBps
500GB 硬盘

实验原理

Cuckoo Hash

散列表是一种查询高效的数据结构，它通过键值与逻辑存储位置的对应关系，来解决存储方案从而降低查询复杂度，使之达到 O(1)的水平。但是哈希冲突会逐渐导致查询性能的下降，故，解决冲突或者降低冲突的发生概率，是一个很重要的研究话题。

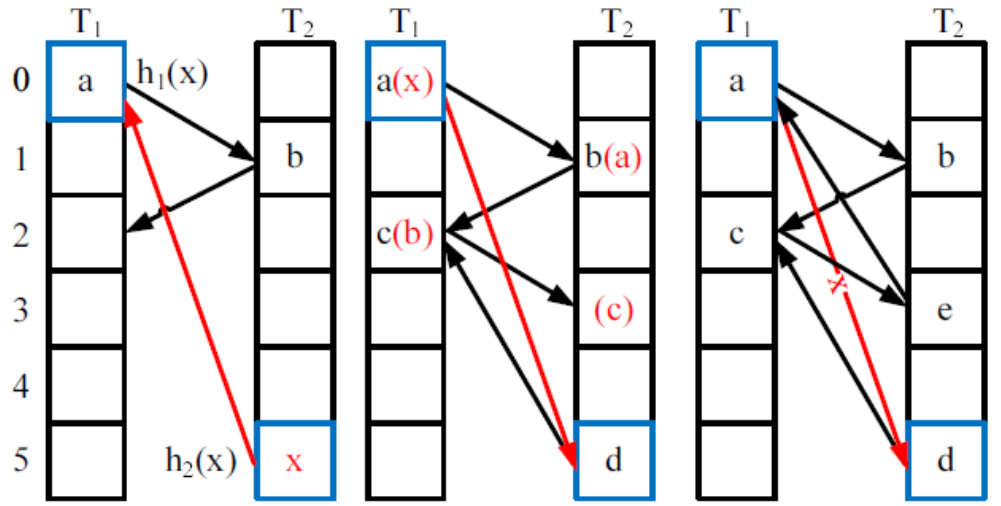
CuckooHash 是一个动态化的静态字典，它以多散列方式解析哈希冲突。

传统 CuckooHash 的设计具有两个哈希表，两个哈希函数。一个插入的元素，分别在两个哈希表中对应有一个实际存储位置和一个备用位置，元素在插入的时候，如果一个哈希函数得到的存储位置发生了冲突，就用另一个哈希函数。两个哈希函数相互独立，且同分布（均匀分布）

CuckooHash 的逻辑结构可以表述为一个有向图。其中，每一条边代表一个插入的元素，这条边连接的两端，起点表示元素的实际存储位置，终点表示该元

素的备用存储位置。

下图是向 CuckooHash 结构的散列表插入元素的演变过程示意图：



(a) Vacant bucket(s). (b) Finite kicks. (c) An endless loop.

一般而言，如果两个哈希函数计算出来的存储位置都是空闲可用的，那么随机取一个（还是取第一个）作为实际存储位置，另一个作为备用位置；如果其中一个被占用（即发生冲突），另一个空闲可用，那么，空闲的作为实际存储位置，被占用的作为备用位置，此时，在有向图中，这个元素所对应的边，就由空闲位置（实际存储位置）指向被占用位置（备用位置）；如果两个哈希函数计算出来的位置都被占用，那么就要随机游走，并进行踢出操作（调解），将已经占用的元素，根据边的指向，踢出到备用空间，迭代这个操作，直至寻找到空闲可用的位置。但是这样就会面临一个问题：在闭环的有向图当中游走，陷入死循环。CuckooHash 必须寻求一个再哈希操作。

Pseudoforest

Pseudoforest Theory（伪森林理论）：一个 pseudoforest 在图论中是一个无向图，子图的每个连通分量中最多只能有一个环，而且这个子图的所有权值之和最大。具有这样的特点：这样的无向图中的边的数目不多于顶点数。

1、伪森林的定义

(a) 一个无向图；

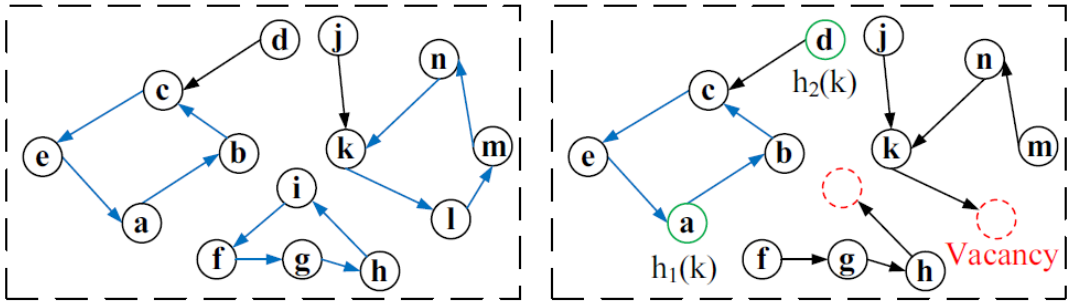
(b) 它的所有连通分量最多只有一个环；

(c) 伪森林的大小取决于图中所有边的边权之和。

2、最大子图

顶点数目瞪圆边数目，它包含一个环，伪森林中任何子图都是最大子图

一个有向伪森林如下图所示：



(a) Maximal. (b) Non-Maximal.

有向图的伪森林中，每个顶点至少有一个出度；

SmartCuckoo Hash

传统的哈希选择一个候选位置作为插入位置，而不考虑该处是否会导致闭环回路，新设计的方案，通过追踪子图的状态去预测插入。

新的方案具有以下特点：

一个元素的插入位置必须属于一个子图，该子图中包含出度为 0 的顶点；

检测空缺是 cuckoo hash 很关键的操作；

每个 bucket 最多存储一个元素，每个元素都有唯一一个备份位置；

哈希表中的个数总是不多于 bucket 的个数，而有向图中顶点的个数也不少于边数，因此，一个子图中最多存在一个闭环，故，用有向图来描述元素位置 SmartCuckoo 是一个有向伪森林；

当一个非最大有向子图插入一个元素时，它将会存储在一个候选位置，一个被踢出的元素存储在有向 cuckoo 路径的最后一个顶点

向一个最大有向伪森林插入元素，不可避免会产生闭环回路

有向伪森林中，除了位于非最大子图有向路径的末端的空间 bucket 是一个出度为 0 之外，其余顶点都有一个出度

最大有向伪森林中，每一个顶点都有 1 的出度，没有顶点能作出踢出操作之后的元素存储目标，也即闭环发生。

因为只有非最大子图才含有空闲 bucket，故，一次成功的插入，取决于元素的候选位置是否有一个（以上）在非最大子图中。

实验内容

环境准备

1、哈希算法的代码实现

SmartCuckoo 源码
https://github.com/syy804123097/SmartCuckoo
libcuckoo 源码
https://github.com/efficient/libcuckoo

2、数据集的准备

(1) Traces and Snapshots Public Archive.

Release Number	Release Date	MacOS dates covered	Homes dates covered
1	July 2014	From June 2011 to May 2014	From September 2011 to May 2014
2	December 2016	From May 2014 to May 2016	From August 2014 to November 2014
1+2 Updated *	December 2017	From June 2011 to May 2016	From September 2011 to November 2014

(2) Bag-of-words data set.

Bag of Words Data Set

Download: Data Folder, Data Set Description

Abstract: This data set contains five text collections in the form of bags-of-words.

Data Set Characteristics:	Text	**Number of Instances:**	8000000	**Area:**	N/A
Attribute Characteristics:	Integer	**Number of Attributes:**	100000	**Date Donated**	2008-03-12
Associated Tasks:	Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	229348

运行代码

编译 libcuckoo 的测试代码

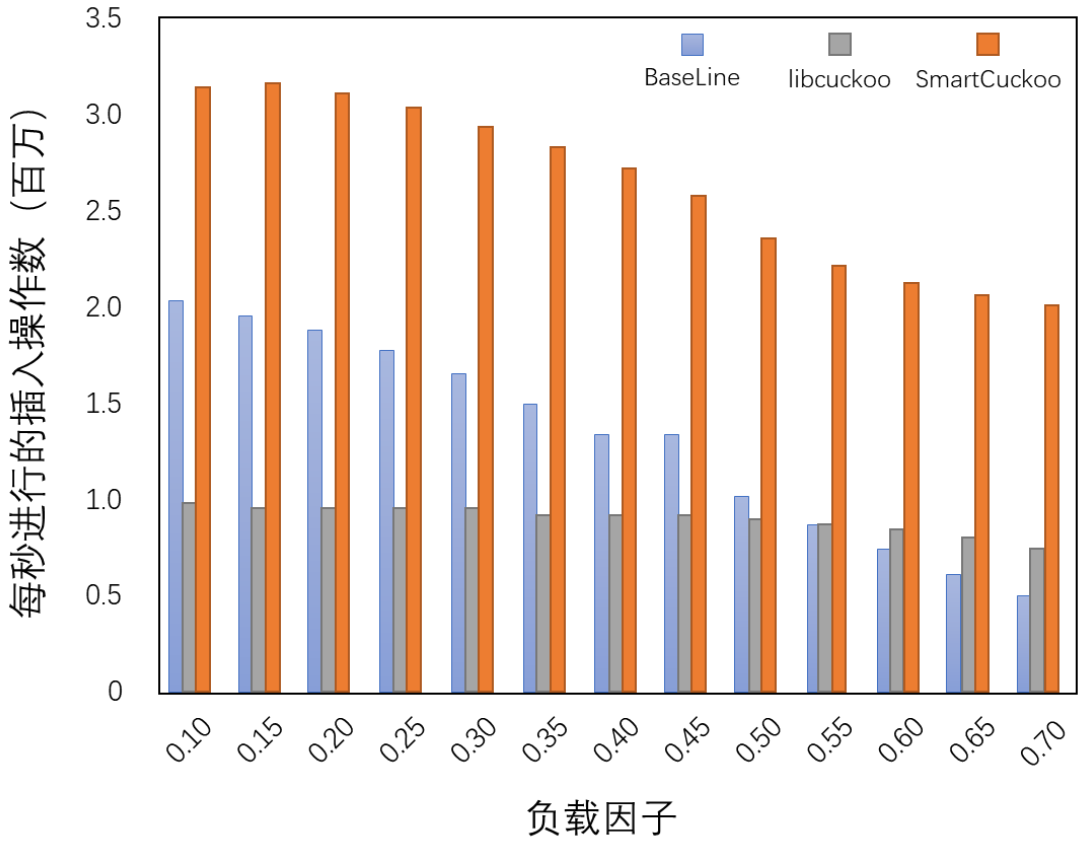
```
root@lxb-laptop:~/smartcuckoo/libcuckoo/build# cmake -DCMAKE_INSTALL_PREFIX=../i
nstall -DBUILD_EXAMPLES=1 -DBUILD_TESTS=1 ..
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for include file pthread.h
-- Looking for include file pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Check if compiler accepts -pthread
-- Check if compiler accepts -pthread - yes
-- Found Threads: TRUE
-- Configuring done
```

编译 SmartCuckoo 的测试代码

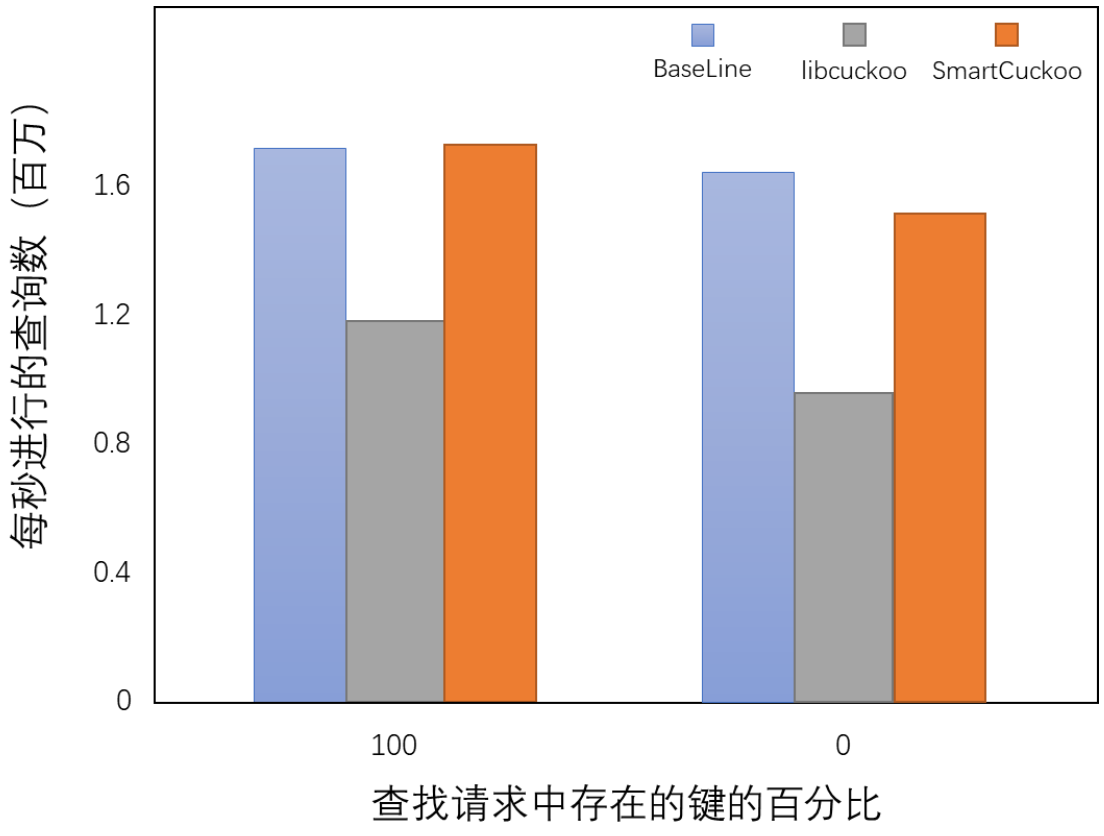
```
root@lxb-laptop:~/smartcuckoo/SmartCuckoo/test# make
g++ -std=c++11 bench_client.cc -I../ ../MurmurHash3.cpp -o ycsb_test
root@lxb-laptop:~/smartcuckoo/SmartCuckoo/test# ./ycsb_test -h
./ycsb_test: invalid option -- 'h'
./ycsb_test [-l trace][-h]
        -l load trace; required!
        -r run trace; required!
        -h  : show usage
root@lxb-laptop:~/smartcuckoo/SmartCuckoo/test# ./ycsb_test -l ./docwords/docwo
d.
docword.enron.txt  docword.kos.txt    docword.nips.txt
root@lxb-laptop:~/smartcuckoo/SmartCuckoo/test# ./ycsb_test -l ./docwords/docwo
d.nips.txt -r ./docwords/docword.nips.txt
=================load phase begin=================
```

实验数据

插入随机数


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
每秒进行的插入操作数（百万）
BaseLine
libcuckoo
SmartCuckoo
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
负载因子

随机数的查找吞吐量


BaseLine
libcuckoo
SmartCuckoo
1.6
1.2
0.8
0.4
0
每秒进行的查询数（百万）
100
0
查找请求中存在的键的百分比

实验结论

快速高效的查询服务是云存储系统的重要组成部分。由于开放寻址的显著特点，CuckooHash 支持快速查询。然而，它遇到了潜在的问题，在项目插入无限循环。SmartCuckoo 是一个新的高效的哈希方案，它的提出，在哈希表中跟踪项目位置，通过使用有向图伪森林来表示散列关系，SmartCuckoo 可以准确预测 Cuckoo 操作和死循环状态，进一步避免随机游走的过程陷入一个无限循环，而是控制在一个 pseudoforest 最大子图。