A

Project Report on ToyDb

Submitted by

Pratik Sanjay Wagh (173050077)

Avais Ahmad (173050043)

Indian Institute Of Technology, Bombay

Sept-2017

Objective

To implement extended hashing over a pf layer and compare its performance with the am layer for a particular file in terms of storage and access cost and also for a given set of queries with a proper interface

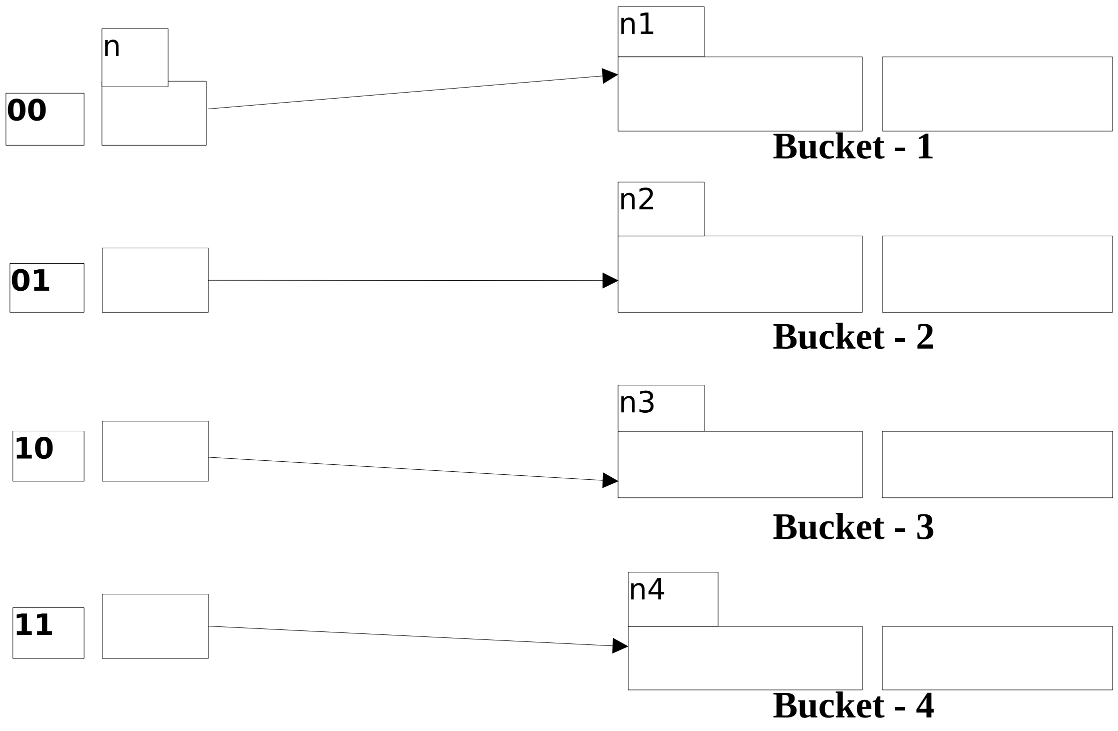
Introduction

Hashing is a technique used to access the location directly without any sequential access or use of index. Extendable hashing is a type of hashing. It uses a tire of buckets for lookup and treat hash as a bit string. Extended Hashing is useful for time sensitive systems because of hierarchical nature and incremental operation.
Toydb basically consists of two layers one is PF layer and other is AM layer. PF layer is a layer which converts the file into a pages and the AM layer provides B+ tree indexing to this pages of a file.

Algorithm

1. Implement extended hashing method on top of PF layer.
2. Calculate its performance on set of queries.
3. Calculate storage and access cost for a given file.
4. Calculate the performance of AM layer on top of PF layer on set of queries.
5. Calulate the storage and access cost for a given file (same as used above).
6. Compare the performance of both the extended hashing method and the AM layer implemented on top of PF layer in both the cases.

Detailed Description of the Implementation



Extended Hashing

Extended hashing is a method to implement dynamic hashing.
In extendible hashing size of hash table changes dynamically according to the inputs given.
Extended hashing achieve this by using Global Depth and Local Depth. Global depth is maintained for the whole hash table , it represent the size of the hash table. Local Depth is maintained for each bucket. It represent the size of each bucket.
Extended hashing does not implement chaining for the overflow. There is only one bucket for each hash table entry.
When the hash table gets filled it splits the bucket. If the local Depth< Global Depth then the bucket will split normally. If Local Depth = Global Depth then we need to doble the size of hash table and the elements of the hash table will be redistributed according to new hash table size.

Doubling of hash table is done using realloc() function. The new table entries created will initially redirect to (new entry – old hash table size). It will point to new bucket once a bucket entry is assigned to it.

Hash function used is:-

(file descriptor+page number) modulo (hash table size)

Initial hash table size is 4
Size of each bucket is 3

Following functions can be performed by Extended hashing Interface:-

1.Insert
2.Delete
3.Find
4.Print Current hash table
5.Hash a given file
6.Exit

1.Insert:- We have to input File Descriptor, Page number and the buffer address . It will insert the entry if it does not exists already. If the bucket overflow it will split the bucket accordingly.

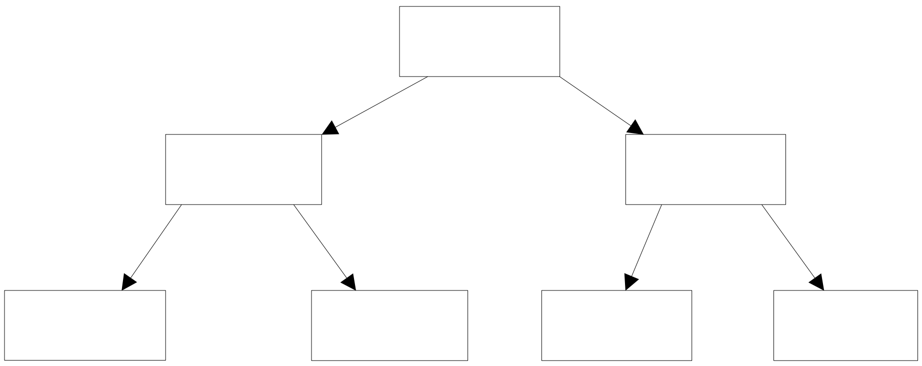
2.Delete:- We have to input File Descriptor and Page number. It will delete that entry from the respective bucket. That space can now be used by a new insersion.

3.Find:- We have to input File Descriptor and Page number. It will find the respective entry and return the buffer address associated with it.

4.Print Current hash table:- It will print the current snapshot of all the buckets of the hash table. It will also mention if the bucket is redirecting to previous bucket.

5.Hash a given file:- It will take “file1” present in the current working directory as input and create a hash table for it.

6.Exit:- This function simply exits from the program.



AM layer

AM layer is implemented at top of PF layer. Its primary goal is indexing it use B+ tree to implemnt indexing.
AM layer is used here for comparision with extended hashing. The performance is compared using timestamp.

Functions of AM layer used are:-
AM_CreateIndex(fileName, indexNo, attrType, attrLength)
This function helps in creating indedx of a file.

AM_InsertEntry(fileDesc, attrType, attrLength, value, reId)
This function helps in inserting entry in the index.

Experimental Results

For a set of queries used for comparison between extended hashing and am layer in terms of storage cost and access cost. We got

1.Time to index a file

1. Insertion time

2. Insertion storage cost

Conclusion

For insertion, searching and finding operation in terms of access cost and storage cost extended hashing is much better than am layer.