Assignment - 1

Database System Implementation

Submitted by:

Deepak Kumar Sood	MT 15013
Neha Rani	MT 15040

Ques 1. Analysis of Extendible and Linear Hashing over different dataset.

Answer: We have used two types of datasets for the analysis:

(a) **Dataset-Uniform**: Contains 100000 uniformly distributed random numbers between the range of 0 and 800000. Numbers may get repeated.

(b) **Dataset-HighBit**: Contains 60000 numbers uniformly generated between the range 700000 and 800000, and 40000 generated between the range 0 and 700000. Numbers may get repeated.

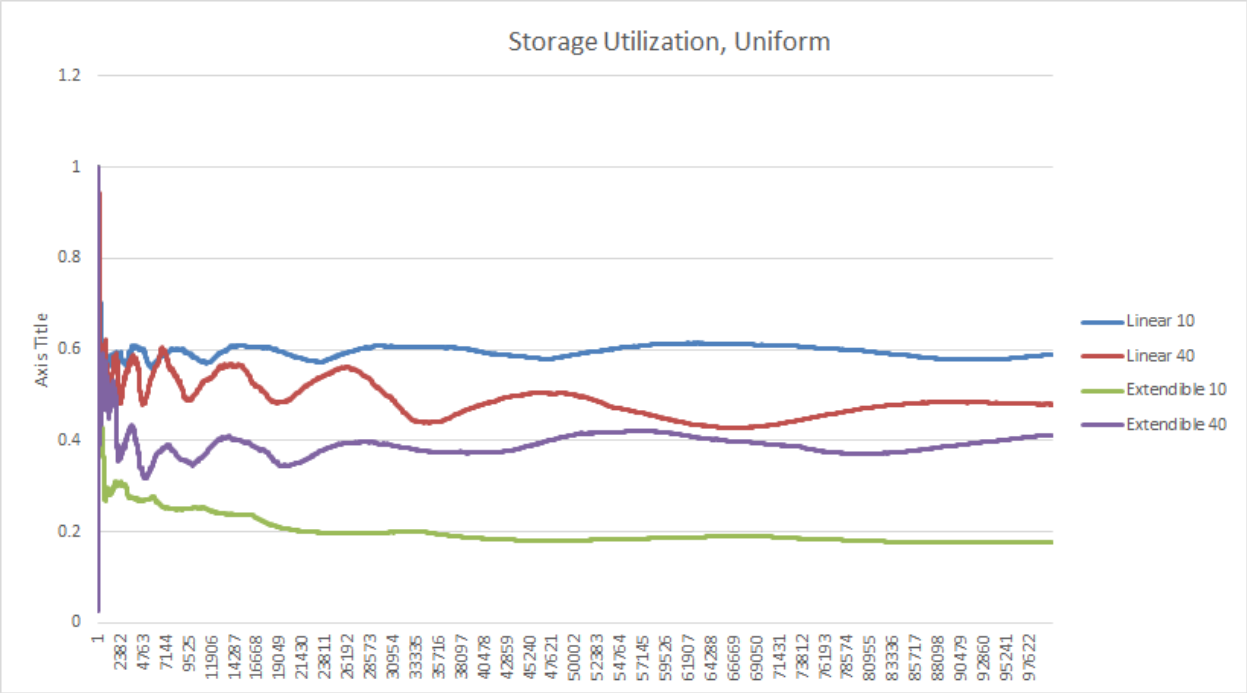
Both Extensible and Linear hashing were evaluated on the basis of the following metrics:

a) **Storage Utilization:** It is a measure of how effectively the available space is utilized. Storage Utilization = No. of records in hash table / (No. of buckets * bucket size)
b) **Average successful search cost =** The number of buckets accessed for a successful search (+1 if the directory is not in the main memory in case of extendible hash)/No. of successful searches
c) **Splitting cost:**
    i) **Linear Hash:** 1 access to read the bucket to be split + k accesses to read k overflow buckets + extra accesses to writ the overflow buckets attached to new and old buckets
    ii) **Extendible Hash:** 1 access to write the old bucket + 1 access to write the new bucket + extra accesses to write the overflow buckets attached to old and new buckets + accesses needed to update the directory pointers if the directory resides on “secondary memory.”

Results:

a) Storage Utilization

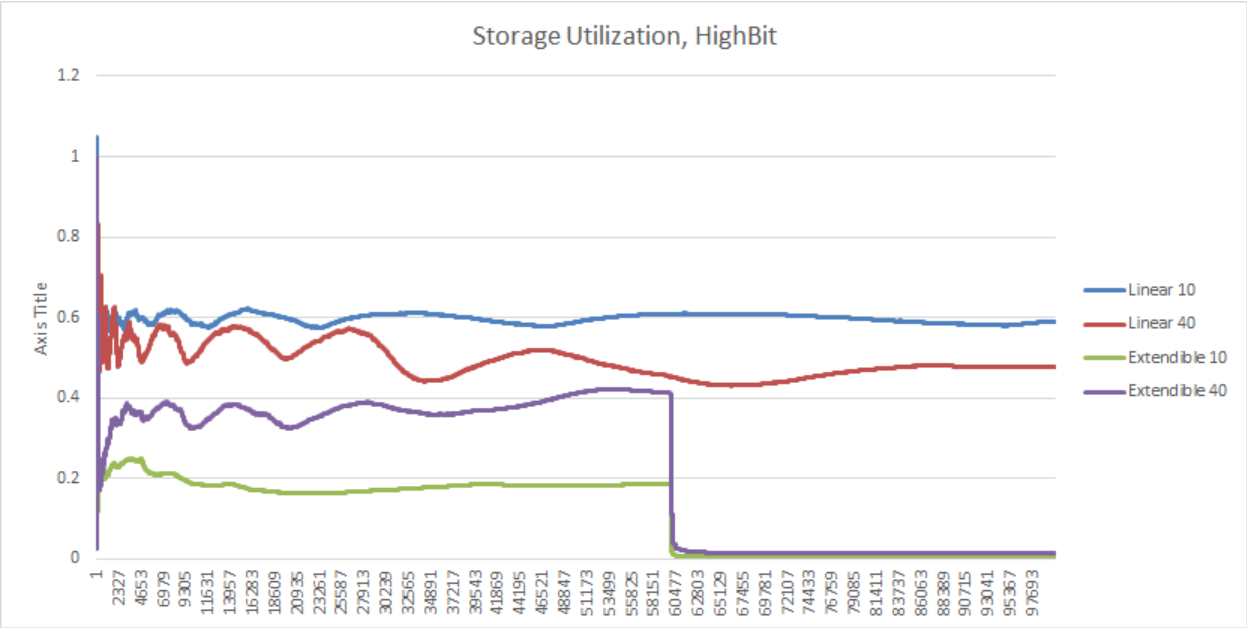
Uniform Dataset:



Observations:

1. Bucket Size increases dramatically at first but it slows down when the data set increases and then almost becomes constant.
2. Bucket size is almost same for both linear hashing and extendible hashing when compared to bucket size 10 and bucket size 40.

Highbit Dataset:

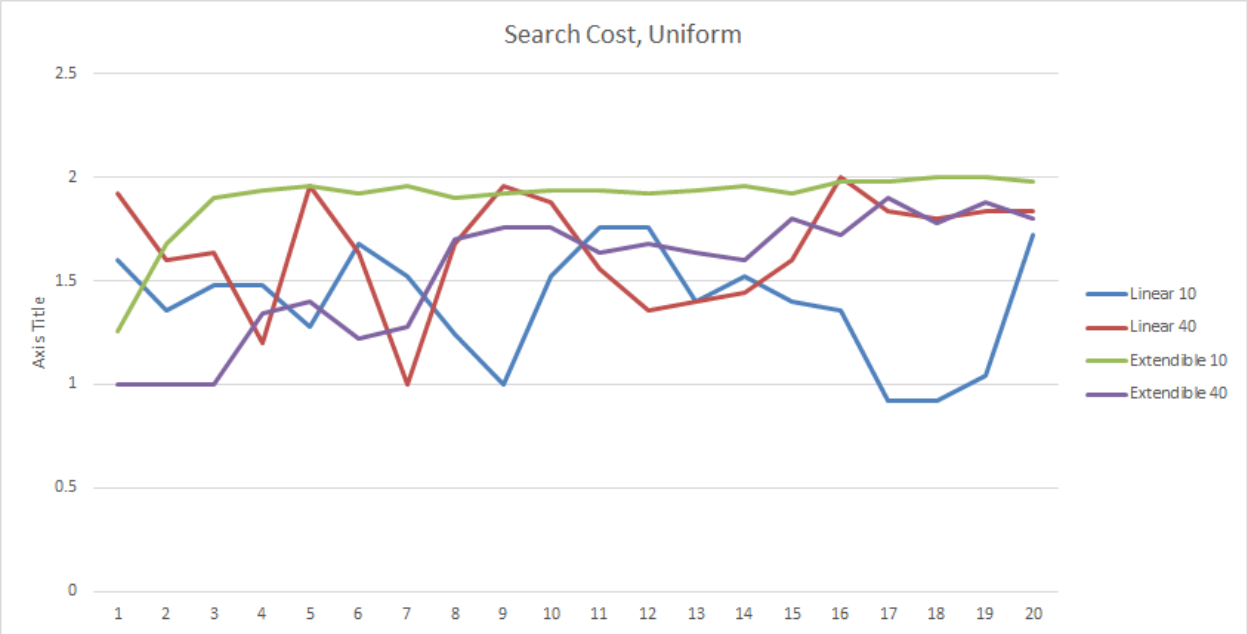


Observations:

1. Utilization falls dramatically at end buckets and most buckets are left unutilized therefore at both bucket size 10 and 40 the plot for extendible falls after 60000 value in x-axis.
2. Linear Hashing show the same behavior as in uniform dataset because the hash function is same as the first and it does not effect.

b) Average Successful Search Cost

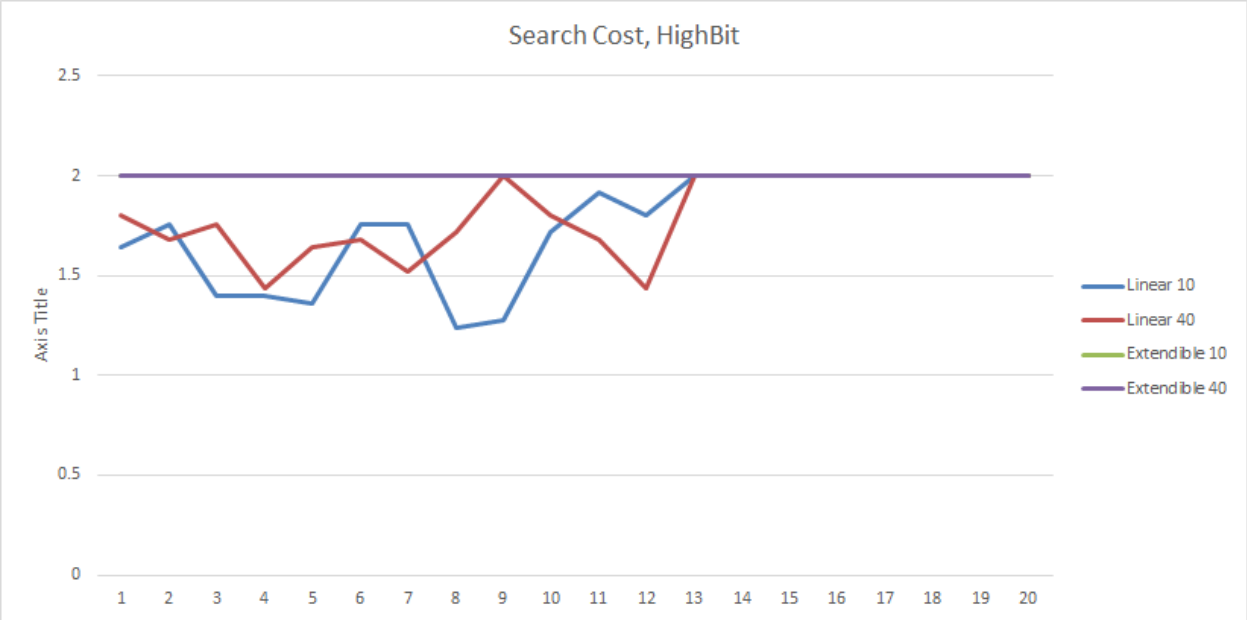
Uniform Dataset:



Observations:

1. Linear Hashing has better cost for successful search as compared to extendible hashing.
2. As clearly seen in plot, extendible is far more efficient than linear hashing in both size of buckets 10 and 40.

HighBit Dataset:



Observations:

1. Search cost for extendible is always 2 for both bucket sizes as buckets are swapped in and out of the secondary memory.
2. Linear Hashing is more efficient than Extendible Hashing in terms of searching at HighBit values.

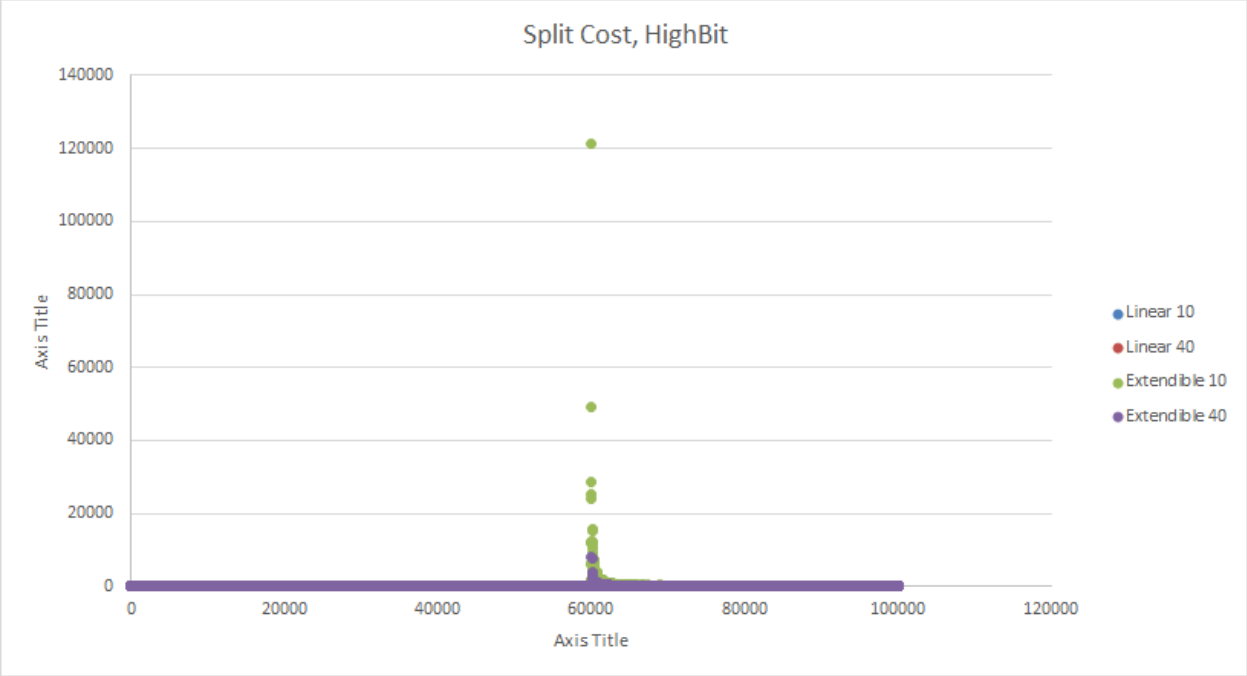
c)Split Cost

Uniform Dataset:

Observations:

1. The number of overflow buckets in Linear Hashing decreases as bucket size increases.
2. Split cost is higher for linear hashing than extendible hashing.

Highbit Dataset:



Observations:

1. The split cost is almost same in case of linear hashing for bucket size 10 and 40.
2. Highbit data is same as uniform dataset in linear hashing as the hash function is same.
3. Split cost for extendible is higher than linear hashing.

Answer 2

a)

- Record Size = 9 +20+ 20+ 1+ 12+ 30+ 10 + 9 + 4 + 4 + 1=120 bytes
- Blocking factor (bf)= floor(2400/120) =20
- Number of disk blocks(b) = ceil(30000/20)=1500

b) Wasted Space = block size-(record size * bf) = 2400-(120*20) = 0 byte

c)

- Transfer rate (tr) =block size/block transfer time = 2400/1 =2400 bytes/ms
- Bulk transfer rate (btr) = (block size/(block size + gap size)) * tr = (2400/(2400+600))*2400 = 1920 bytes/ms

d) Average number of block accesses = b/2 = 1500/2 = 750 blocks ; if record is found
= 1500 blocks ; if record is not found

e) av. time to search an arbitrary record = seek time + rotational delay + (b/btr)* no of blocks
= 20 + 10 + (2400 / 1920) * 750 =967.50 ms
ii. if n=b: time = 20+10+(1500*(2400/1920)) = 1905 msec = 1.905 sec

f) av. time to search record which is scattered = no of blocks * (seek time + rotational delay + block transfer time)
= 750 * (20 + 10 +1) = 23250 ms

g) Time to search using binary search
= (seek time+ rotational delay + block transfer time) * $\log_2$(no of blocks)
= ceiling (log 2 b)*(s+rd+btt)
= ceiling(log 2 1500)*(20+10+1) = 46.5 sec