b-trees

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
Midway check-in:
Two design docs today (Canvas) & tests on Sunday

Next weekend: Lab marathon for midway check-in & tests will run every 1 hour
(similar idea to OH for midterm preparation last weekend)

Remember: Midway check-in goal is not to test you but to “force” you to see the scope and size of the project if you are behind and get you going
**Midway check-in:**
Two design docs today (Canvas) & tests on Sunday

Next weekend: Lab marathon for midway check-in & tests will run every 1 hour (similar idea to OH for midterm preparation last weekend)

Remember: Midway check-in goal is not to test you but to “force” you to see the scope and size of the project if you are behind and get you going

**Midterm 1:**
Grading out in about 10 days - come to OH for 1-1 feedback + 1-2 sections

Remember: Midterm goal is not to test you but to *force* you see the concepts and way of thinking if you have fallen behind.
Midway check-in:
Two design docs today (Canvas) & tests on Sunday

Next weekend: Lab marathon for midway check-in & tests will run every 1 hour (similar idea to OH for midterm preparation last weekend)

Remember: Midway check-in goal is not to test you but to “force” you to see the scope and size of the project if you are behind and get you going

Midterm 1:
Grading out in about 10 days - come to OH for 1-1 feedback + 1-2 sections

Remember: Midterm goal is not to test you but to *force* you see the concepts and way of thinking if you have fallen behind.

Research Meeting 2:
Friday 5pm, New research on tree-based concurrency control
Scaling to many many cores. My office and Zoom. No recording. Email me if you want to participate by Thursday.
filtering data: point/range queries/locate data to update

index knows order about the data
initial state
columns in
insertion order

sorted A  B  C

propagate
order of A
**select** max(D), min(E) **from** R where (A>10 and A<40) and (B>20 and B<60)

avoid scan of A
avoid TR on B
work on a **restricted area**
across all columns
good for memory hierarchy
today+1: more about indexing

tree indexes in data systems
primary/clUSTERED (ALL cOLUMNS)

SECONDARY/UNcLUSTERED INDEXES
subset of columns

you are doing both in the project
Btree on A, A is sorted, order is propagated to the rest of the columns

clustered index on A
(no need for mappings)

every table can/should have one (be a) clustered index
A B C

Btree on C, copy of C is sorted, we keep a copy of the positions that map on the clustered index

pos C
7 8 10 1

pos B

secondary index on any column(s) needs positions
original order

A B C

clustered index/base data

A B C

secondary index

pos C

7 8 10 1

clustered index case plan vs secondary index plan

\[
\text{select } \max(B) \\
\text{from } R \\
\text{where } A < 20
\]

\[
\text{select } \max(B) \\
\text{from } R \\
\text{where } C < 20
\]
A
B
C

original order

A
B
C

clustered index/base data

A
B
C

secondary index

pos C

7
8
10
1
5

same discussion as with sorting
“It could be said that the world’s information is at our fingertips because of B-trees”

Goetz Graefe
Microsoft, HP Fellow, now at Google
ACM Software System Award
ACM SIGMOD Innovations Award
b-tree = a “fat” tree, each node holds >>1 keys

Why is this good?
ok and how do we build, search, update a tree efficiently?

structure=complexity=if statements, random access, instruction misses, etc.
= no free lunch
1,2,3…  12,15,17  20,…  ...

sorted array

page size: 64K - holds 16K 4 byte ints
N elements, P pages
The diagram shows a sorted array with elements 12, 20, 35, and 50. There are two conditions mentioned:

- Elements less than 12 are in the range 1, 2, 3, ...
- Elements greater than or equal to 12 are in the range 12, 15, 17, ...

The array is divided into pages, with each page holding 16K 4-byte integers. The page size is 64K, which holds 16K elements. The diagram also indicates that there are N elements and P pages.
info to navigate lower level value-pointer

page size: 64K - holds 16K 4 byte ints
N elements, P pages

sorted array

12,20  35,...  50,...

<12   >=12

1,2,3...  12,15,17  20,...

4+4 bytes for each page (value+pointer)
64K/8= index 8K pages
A sorted array is divided into pages, each holding 16K 4-byte integers. The page size is 64K, allowing for 8K page indices. Each page can contain 4+4 bytes of information: a value and a pointer. The diagram illustrates how to navigate between levels, with keys 12, 20, 35, and 50, and pointers to subarrays (1, 2, 3,...) and (12, 15, 17, 20,...). The next level can index 8K pages.
The diagram illustrates a tree structure with the following components:

- **Root**: A node labeled with a range of values (30,50).
- **Fanout**: The root node has a fanout that distributes values to its children.
- **Internal Nodes**: The internal nodes contain ranges of values (12,20, 35,..., 50,...).
- **Leaves**: The leaves contain ranges of values (1,2,3..., 12,15,17, 20,..., ...).

This structure is typical of a B-tree or similar data structure.
The diagram illustrates a tree structure with internal nodes and leaves. The root node is labeled as 30, 50, and there are internal nodes with labels 12, 20, 35, 50, etc. The leaves are labeled 1, 2, 3, ..., 12, 15, 17, 20, ..., with an ellipsis indicating more leaves. The height of the tree is given by $\log_{\text{fanout}} N$, where $N$ is the number of leaves. The fanout property indicates the number of children each internal node can have.
random accesses

height $\log_{\text{fanout}} N$

root

fanout

internal nodes

leaves

random accesses

height $\log_{\text{fanout}} N$

root

fanout

internal nodes

leaves

$1,2,3\ldots$ $12,15,17$ $20,\ldots$ $30,50$ $35,\ldots$ $50,\ldots$ $\ldots$
how do we search the leaves?
a) sorted and b) unsorted leaves
should we store leaves as independent nodes or as a single contiguous column
should we store leaves as independent nodes or as a single contiguous column

diff in structure of clustered vs secondary index
1) compare tree search to binary search cost
   assume a clustered index

DATA DATA DATA DATA
vs
DATA DATA DATA DATA

assume: input 1 Billion integers array, page size 64Kb, system 64bit

2) design a cache conscious tree
   (a tree that minimizes cache misses)
Browse: **textbook:** Chapters 8, 9  
(general indexing & H/W)

Read: **textbook:** Chapter 10  
(b-trees)

Read: **Modern B-Tree Techniques**  
by Goetz Graefe  
Foundations and Trends in Databases, 2011  
Sections: 1,2,3,5

Read: **Making B+trees Cache Conscious in Main Memory**  
Jun Rao and Ken Ross  
ACM **SIGMOD** International Conference on Management of Data, 2000
b-trees

DATA SYSTEMS

prof. Stratos Idreos