b-trees

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
I spend a lot of time debugging. Am I doing something wrong? Maybe, but probably not.

1. Learn to use gdb
2. After spending X time debugging, ask for help
3. Enjoy it :)

Test extensively every few lines of code. Isolate problems - one change at a time.
grading adapts to your starting level and progress
midterms
how to prepare

open book, notes, no laptop/discussion
material from lectures only
check all quizzes and questions

quiz-like questions - no exact answer

expectations: describe the design space - chose what you think is the best approach (>1 if we ask for it) and then analyze in detail all requests - if you made the wrong choice in the begging it is ok - but say so if you find out in the end and explain as much as possible

explain all steps and tradeoffs
midterms
how to prepare

open book, notes, no laptop/discussion
material from lectures only
check all quizzes and questions

again office hours this weekend
for discussion on concepts and project
Sunday 3-5pm
filtering data: point/range queries

index knows order about the data
but wait, why not just sort the data (array) + binary search?

why bother with creating/maintaining another data structure?
1 pass to sort each page (2N pages)

$2N(\log_2(N) + 1)$
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
space overhead - update overhead - which ones to build?
today+1: more about indexing

trees indexes in data systems
“It could be said that the world’s information is at our fingertips because of B-trees”

Goetz Graefe
HP Fellow
clustered (all columns)

index

data

secondary indexes
subset of columns
Btree on A, A is sorted, order is propagated to the rest of the columns

every table can/should have one (be a) clustered index
Btree on C, copy of C is sorted, we keep a copy of the positions that map on the clustered index

Secondary index on any column(s) needs positions
select C from R where A<x
select A from R where B<z
select D from R where A<x and B<z

which indexes to create
declarative interface
ask what you want

indexes/views/tuning knobs

but ... db cracking, adaptive* ideas

db system
how to use a b-tree index (plans)?
(same discussion as we did for sorting!)
clustered index case plan vs secondary index plan

`select` max(B) from R
where A<20

`select` max(B) from R
where C<20
ok and how do we build, search, update a tree index efficiently?

structure = complexity = if statements, random access, instruction misses, etc. = no free lunch

node size, data organization fanout ...

DATA DATA DATA DATA DATA DATA DATA
column A, select A<20

design a tree index and compare search
by traversing tree vs binary search costs?

assume: input 1 Billion integers array, page size 64Kb, system 32bit
consider node size/structure, fanout, bytes touched
page size: 64K - holds 16K 4 byte ints
N elements, P pages

sorted array
Page size: 64K - holds 16K 4 byte ints
N elements, P pages

1,2,3… 12,15,17 20,…... 12,20 35,… 50,…

info to navigate lower level value-pointer

<12  >=12

sorted array
Sorted array

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

<12, >=12

4+4 bytes for each page (value+pointer)
64K/8 = index 8K pages

Info to navigate lower level value-pointer
info to navigate lower level value-pointer

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

sorted array

1,2,3… 12,15,17 20,… ...

<12  >=12

12,20 35,… 50,…

4+4 bytes for each page (value+pointer)
64K/8= index 8K pages

can index 8K pages of the next level

30,50
The diagram illustrates a binary search tree (BST) with the following key points:

- **Root**: The topmost node with the values 30 and 50.
- **Internal Nodes**: Nodes with intermediate values, such as 12 and 20.
- **Leaves**: Nodes at the bottom with values like 1, 2, 3, and 12, 15, 17.
- **Fanout**: The branching out of nodes from the root, indicating the structure of the tree.
- **Value Ranges**: Each node contains value ranges, for example, 12, 20 for the internal node.

The tree structure shows how data is organized hierarchically, with each node potentially having two children.
height \log_{\text{fanout}} \sqrt{N}
random accesses

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

root

fanout

internal nodes

leaves

1,2,3...

12,20

30,50

35,...

50,...

20,...

12,15,17

1,2,3...

24/28
how do we search the leaves?

get 15
get 15-25

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

textbook: Chapter 10 (b-trees)
textbook: Chapter 13 (qe)

next time: more update b-trees & robust access patterns
b-trees

DATA SYSTEMS

prof. Stratos Idreos