B-trees

10-30-2003

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the assignment?
- Group meetings and final format.

Some Details on Disks

- As you are well aware, your computer has both internal memory and disks. Typically the disks are much bigger (storage wise), but they are also MUCH slower.
- The disk can have multiple platters that spin with read/write heads that hover over (and under) the platters.
- One ring on a platter is a track, a stack of tracks in a cylinder. Cylinders are borken into pages.

B-Trees

- The biggest difference between a B-tree and the binary search trees we have looked at is the branching factor. A B-tree will typically have hundreds or thousands of children under a given node.
- The reason for this is that a node should fill a "page" on disk. This optimizes the disk operations. B-tree applications typically don't fit in memory.

Requirements of a B-Tree

- Each node has: a number of keys, the set of keys stored in non-decreasing order and a boolean saying if it is a leaf.
- All internal leaves also have children, one more than the key count. The fall "between" the keys in that node and the keys in the children are "between" the parent keys.
- All leaves have the same depth.
- There is a minimum degree, t. All nodes (except the root) must have at least t-1 keys and at most 2t-1 keys.

Searching a B-Tree

- The procedure of searching a B-tree is quite simple. It is like a binary tree, except that at each node we have to walk though multiple keys to find either the one we want, or the two that bound what we want.
- If we find two that bound what we want we have to load in the data for that child from disk and recurse to that child.

Inserting into a B-Tree

- As with inserting into a binary tree, an insertion to a B-tree can be done with a single pass down through the tree that works much like a search. The difference is that now we aren't adding a new node at the bottom. Instead, we add the key in a leaf.
- The problem is when a leaf is full we can't add more. So we split.

Splitting

- The actual procedure we use is that as we go down, any time we see a full child, we split it. This way we do things in one pass.
- To split a node, we add the median key in it to its parent and turn it into two children of that parent, each with half as many nodes.
- When the root is full we split it by creating a new root.

Minute Essay

- Combine what you know about direct access files with what we just talked about for a B-tree. Give a brief explanation of how you would implement the internal code of a B-tree. How is it different from what you have been doing with our other trees?
- Remember to do a group evaluation for assignment #3.