EECS 405 - Project Progress Report

Nathan McKinley (ndm25) and Umang Banugaria (uxb3)

April 9, 2013

We are performing a variant of project 1.

We will design and implement a system with a graphical user interface that allows the user to manipulate parameters of a load on a tree-structured index. The parameters will be:

a) size of index record
b) size of data record
c) size of index pointer
d) size of data pointer
e) block size
f) range and type of keys
g) percentage of read/insert/delete operations
h) number of steps of the simulation
i) whether to perform coalescing as specified in B-tree and B+tree definitions.
j) the amount of main memory on the system

The goal of this project is to explore how the above parameters can affect the performance of B-trees/B+trees. In particular, we would like to do experimentation to find the conditions under which coalescing underflowed nodes could be beneficial or detrimental to the overall performance. For the experimentation, we will perform random operations as specified in g) for a number of steps as specified in h). To measure the performance, we will display the number of disk blocks loaded over the whole simulation, the height of the tree, the overall load of the tree, and the tree itself. We will do this in a GUI.

One of the challenges for this project may be in identifying the different parameters or combination of parameters that can cause coalescing to be beneficial/detrimental. However, most of the testing can be scripted allowing for quick testing of many different parameter values.

For the implementation of the project, we will work primarily in python, using the standard python GUI toolkit (Tkinter). Nathan will be responsible for the simulation, and Umang will be responsible for the tree representation. For the trees, we will be using the standard B-tree / B+tree insert, delete, and lookup functions.

At this point in time, the B+ tree representation has been completed and tested. The B+ tree was created to allow the modification of the above parameters and uses dense indexing for the first level of the index. Work has started on the implementation of the B tree and will also allow the same insert, delete, and lookup functions and customizations of the above parameters.

Currently, the GUI framework is set up and capable of displaying the B+ trees. The interface should be almost identical for the B trees when they are completed. It is also capable of receiving user input through a menu bar or through input boxes (both are currently being tested for usability). The GUI is based heavily on TreeView from the NLTK toolkit, which uses TKinter to display trees.

For the remaining timeline of the project, we plan to have the following components completed by the specified date:

1. the B tree implementation and testing will be completed by April 14
2. the B-tree integration into the GUI will be completed by April 15

3. the simulation runner will be completed by April 17
4. the tests to be included in the report will be run and completed by April 18
5. the remaining days will be reserved for writing up the report

Our references will be simply our class notes and textbook. Since this project is for exploring what parameters of the index will affect the empirically-optimized decision of avoiding coalescing underflow nodes (which is usually chosen in industry), we are not using advanced algorithms. Nonetheless, exploring the conditions under which tree-structured indices should or should not be coalesced is interesting.