Hash Table vs. AVL Tree

A hash table is better than an AVL tree in most situations dealing with large amounts of data because hash tables, when implemented correctly have a big-O(1). AVL trees have a big-O(log n) for inserting and searching when implemented correctly. We could see that the AVL Tree implementation ran a lot slower than the hash tables through our timed test of parsing the data. By using the hash table we were able to parse 80,000 documents in about 3 minutes and 30 seconds whereas our AVL parse was timed at 9 minutes 40 seconds. Using a small xml file with only seven documents, the AVL tree averaged about 0.285 seconds, while the hash table averaged about 0.28 seconds. This data shows that hash tables are more efficient when working with large amounts of data but are about the same efficiency as AVL Trees when dealing with smaller data.

We also increased our search and insert time by making our hash table include AVL Trees. Our hash table consists of 1299827 AVL trees. This made all of the AVL trees very shallow, which makes traversing them a lot faster.

For smaller data sets the AVL tree is faster than the hash table because the hash function slows the inserting and searching down. With big files the hash table works a lot faster because it traverses shallower trees whereas the AVL Tree would have a large height to traverse each time the insert or search method was called.