|  |  |  |  |
| --- | --- | --- | --- |
|  | **stage\_2\_c** | **stage\_3\_b (fanout 10)** | **stage\_3\_b (fanout 200)** |
| **process\_record\_1** | 0.001082 | 0.012 | 0.051 |
| **process\_record\_2** | 0.020904 | 0.095 | 0.161 |
| **process\_record\_3** | 0.022236 | 0.069 | 0.078 |
| **process\_record\_4** | 0.027764 | 0.033 | 0.055 |

There is a discovery with this, yes. It appears that the B+ tree takes longer to do all these queries due to the fanout values. Since we have more nodes within the leaf, we have to keep looking before moving on to the next leaf.

As for the effect of *n*, the larger it is, the more files we Java needs to open to generate the data. Of course, since we need to generate the user tree, it’s instantly a linear operation. Performing queries for this would end up being in quasilinear. On an unrelated note, since we changed from C to Java between stage\_2\_c and stage\_3, the time taken will for sure be different due to the limitations and performance of Java.