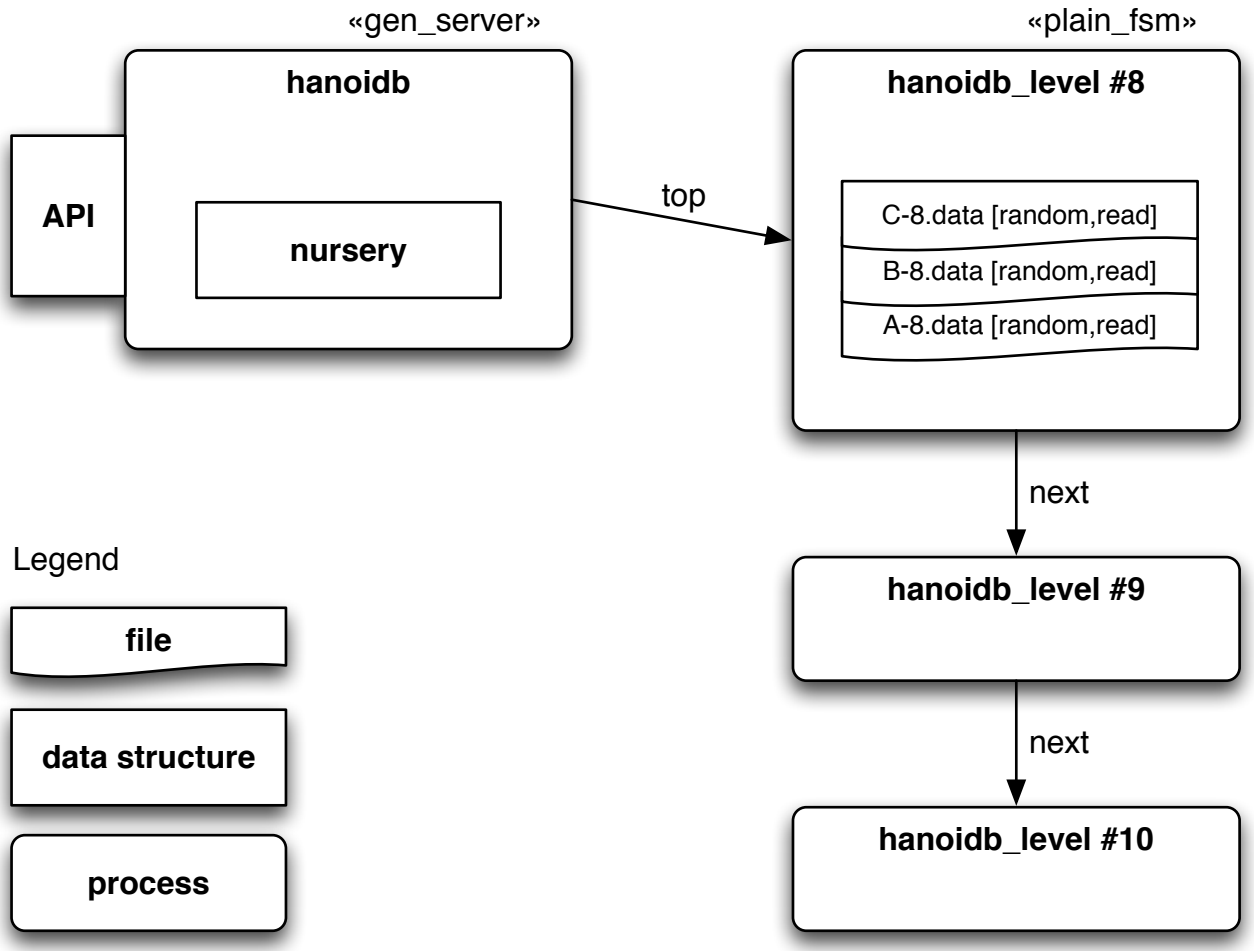
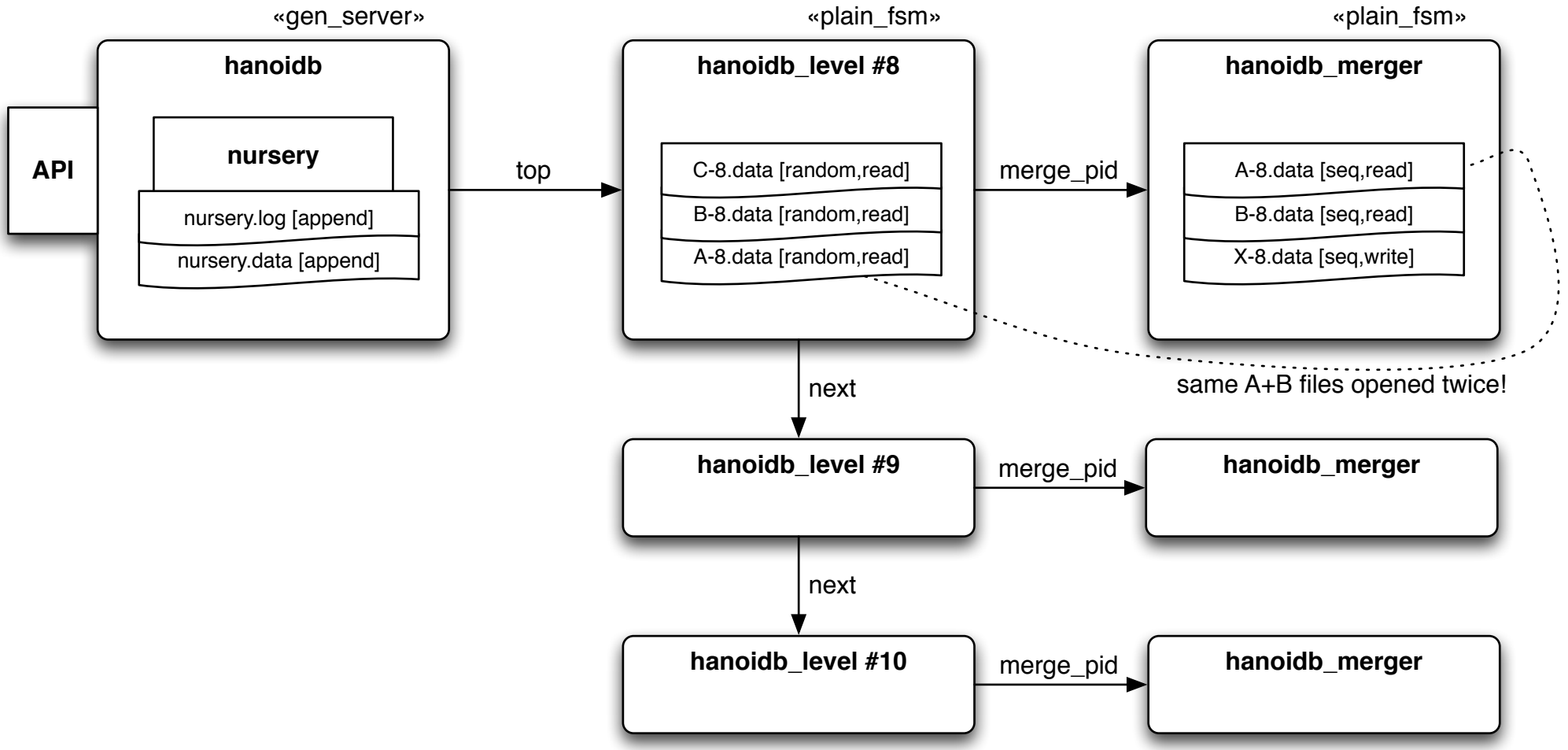
get: Looks up a given key in the nursery (recently added), if not found send a one-way message that flows down the levels, inspecting files in the order C, B, A (newest to oldest). When found, the level replies directly using gen_server:reply/2. Thus, there can be multiple get requests flowing down the levels concurrently (but only one being considered at each level at any one point in time). Doing a file-level lookup first uses a bloom filter to avoid unnecessary tree walks.

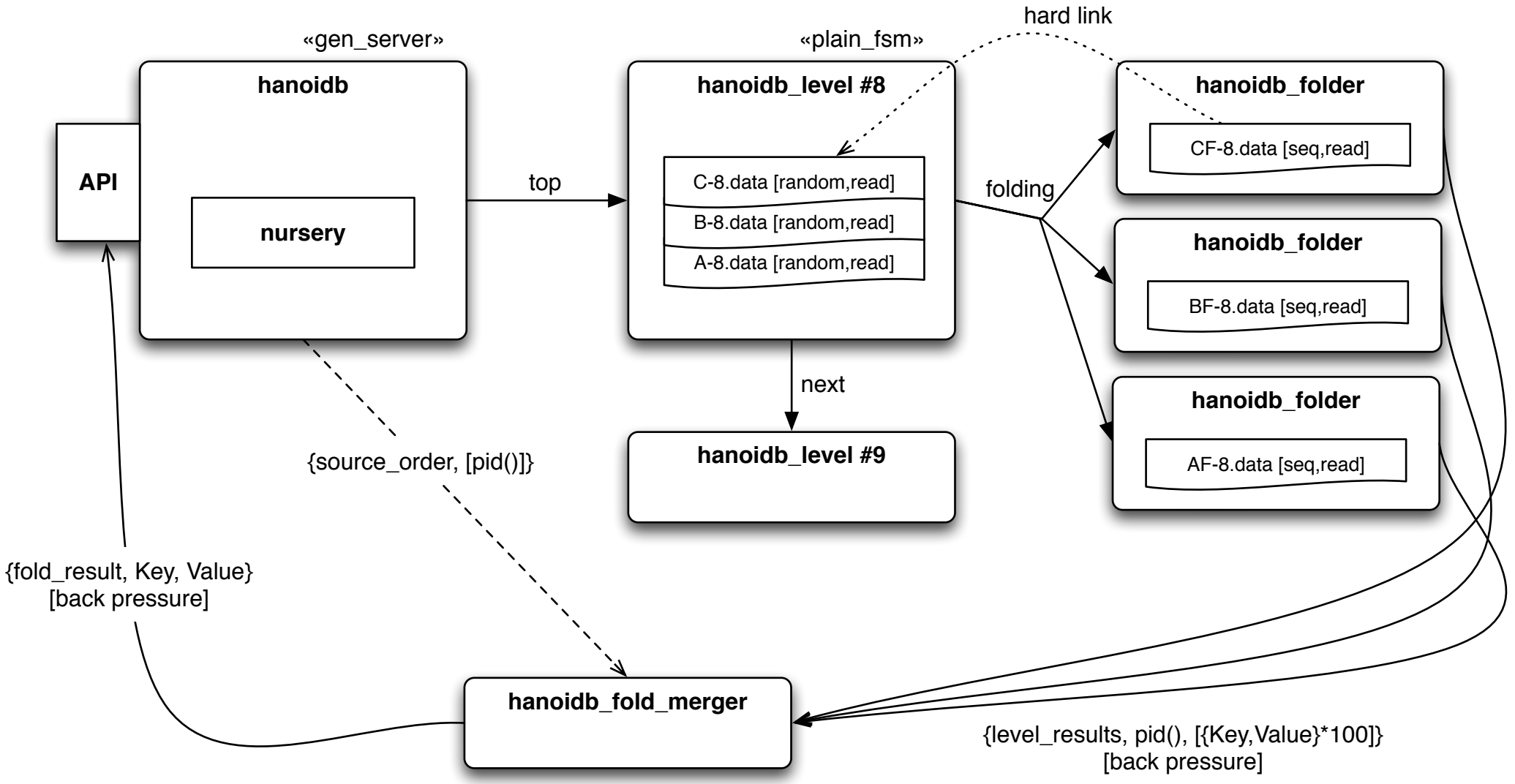
Data files at level #n contain at most 2^n key/value pairs.



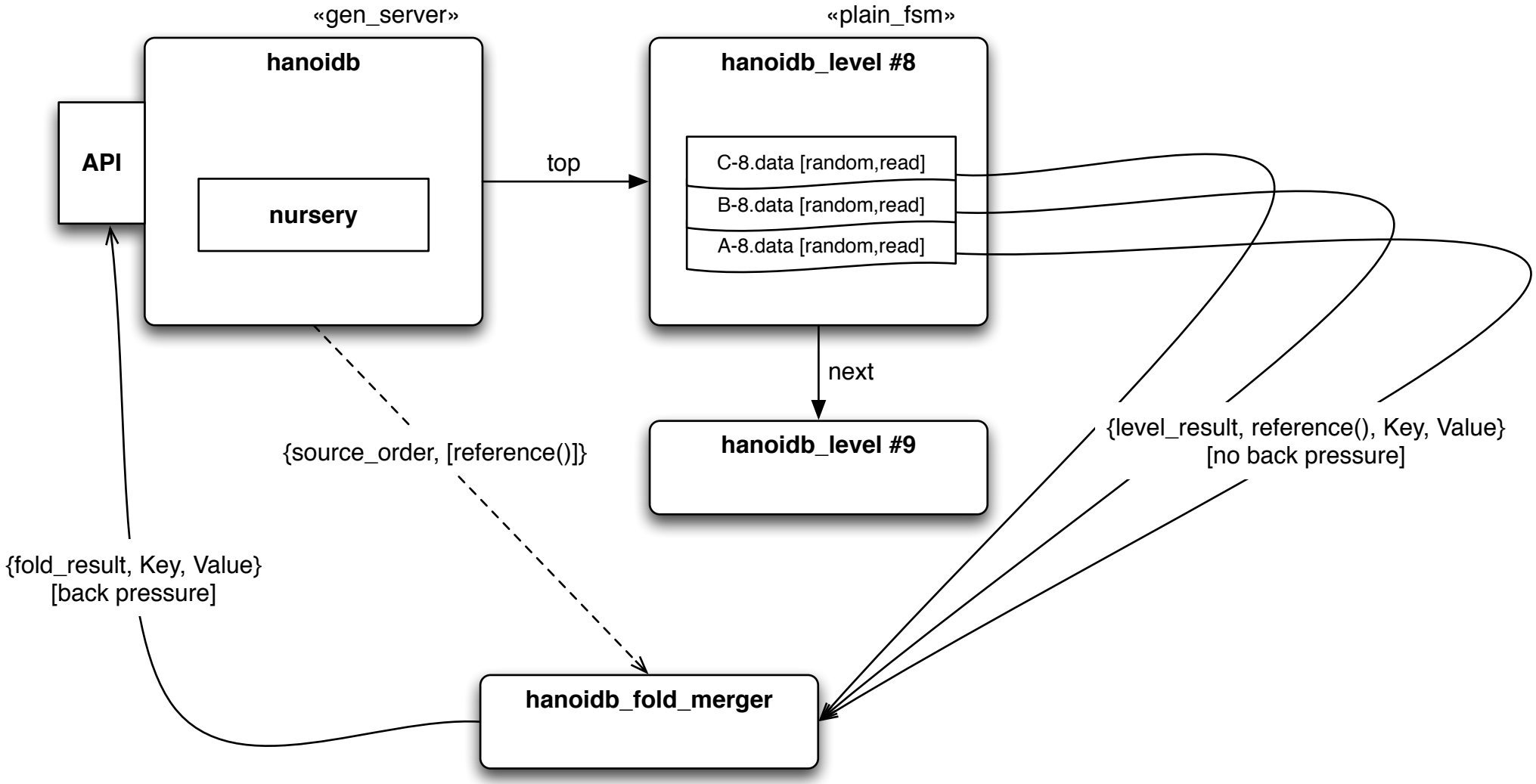
put: Normal insertion just adds data to in-memory nursery gb_tree, and appends {K,V} to nursery.log. When nursery is full (256 entries) the data is written to the (b-tree) nursery.data. This file is then closed and "passed" into the top level (the level will rename it to A, B, or C). Finally, an "incremental merge" is issued, which will flow down through the levels, and instruct the merge processes to each do 256 "steps". Independently, when a merge process completes, the corresponding level N will "pass" the resulting X-*N*.data file to the level *N+1*, A and B files are deleted, and C is renamed A. The 256 "steps" guarantees that when an inject happens, the target level does not have a C file, i.e. there is room.



snapshot fold: all data files are hard-linked; i.e. A-8.data becomes AF-8.data, and are scanned concurrently (one process per data file). When all folder processes have been started, the fold_merger is told the *source order* in which to consider the individual fold results, after which it does selective receive when running out of values from a given folder. The fold_merger merges such data from *all levels.*
The current implementation only allows one concurrent snapshot fold to avoid unlimited number of open files. Further snapshot folds will be enqueued and concurrent put/gets will be services all along.



blocking fold: [only when limit < 10] All data files [A,B,C]-N.data are scanned sequentially for up to *limit* matching entries. The fold_merger is told the *order* in which to consider the individual fold results, after which it does selective receive to grab fold results. The fold_merger merges such data from *all levels.* Thus, all potential results are in the fold_merger's inbox before merge commences. Currently only used in riak's list buckets.



HanoiDB File Format