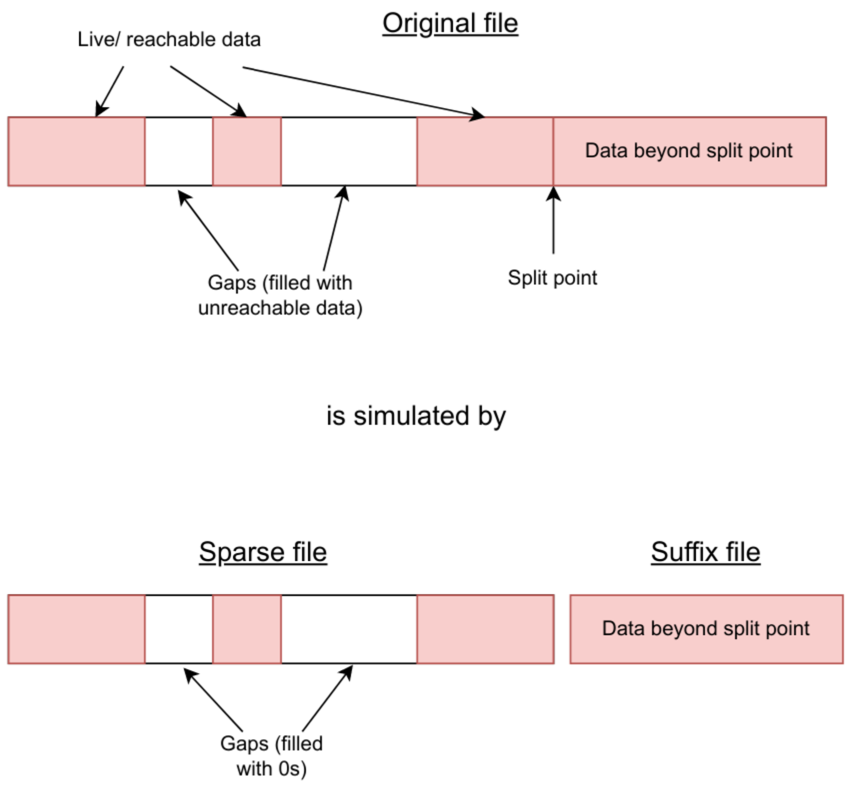
Irmin layers/GC: summary of main steps

Background: The current design of Irmin layers/GC involves splitting an (append-only) "pack store" file into two parts: a "sparse file", consisting of the live parts of the file before the split point, and the "suffix file", consisting of everything after the split point.



When performing a GC, we construct a completely new sparse+suffix, and then switch the running implementation from the old versions (of the sparse+suffix) to the new versions.

Summary of main steps:

Step abbreviation	Description
main-start-gc	For a given commit ("gc commit"), start GC by launching a worker process. The worker's job is to construct the **next** new versions of the sparse and suffix file. The new split point is the location of the given commit. While the worker runs, the main process continues to append to the **current** suffix file. When the worker finishes, the main process switches to use the new sparse and suffix.
w-start	Worker starts executing.
w-create-reach	Worker calculates the reachability data (which parts of the file are "live") for the given commit. This data is written to a file ("reachable.tmp" or similar). The format of the file is a sequence of integers. Each pair of integers is an (offset,length) pair ("the data at this offset, with this length, is live"). To avoid any memory overhead when creating this file, the data is "raw", in the sense that there can be multiple entries for the same offset, and the data is not sorted. Typical file size for reachable.tmp is 5GB.
w-create-sorted	Worker sorts the data in reachable.tmp by offset, and places result in a file ("sorted.tmp" or similar). To use minimal memory, the data is sorted on disk, using an "external sort" routine. Typical file size for sorted.tmp is 5GB.
w-create-extents	The data in sorted.tmp is read through linearly to produce a much smaller "extents.tmp" file. Adjacent regions are combined into a single region. Regions which are "almost adjacent" are also combined. A region whose end is within "gap_tolerance" bytes of the start of the next will be combined with the next region. The aim is to reduce the number of regions so that the resulting extents file is small enough to be comfortably held in memory. Typical file size for extents.tmp is 35MB (?)
w-create-sparse	At this point, the worker knows exactly the live data it needs to create the new version of the sparse file. The new version of the sparse file is created. This uses directly extents.tmp to construct the sparse data file and the sparse map file. Typical size for sparse.data: 5GB. Typical size for sparse.map: 45MB (?).
w-create-suffix	The worker now creates the next version of the suffix, consisting of all the data from the new split point onwards. This is currently done by creating a new suffix file and copying from the old suffix file. As a result, the live data from the split point (5 cycles' worth in Tezos usecase) is duplicated. Typical case of "suffix.next": 30GB
w-finish	Worker terminates.
main-detect-worker-term	Main process detects the worker has terminated. It blocks to perform the next steps.
main-load-next	Main loads the next versions of the sparse and suffix. This is as simple as opening a few files, so should not take much time.
main-patch-suffix	Main copies any further data from the end of suffix.current to the end of suffix.next (suffix.current is live after the worker terminates, so could have additional data appended that is not in suffix.next). Typically the data copied is a few MB or so, so this should also not take much time.
main-switch-to-next	The next versions of the sparse and suffix are made "live".
main-cleanup	The old sparse and suffix are deleted.
main-continue	The main process continues executing as before, but with the new sparse and suffix.

Note: in the above we have not discussed when calls should be made to fsync.