WRITEBACK THROTTLE

Previously, the filestore had a problem when handling large numbers of small ios. We throttle dirty data implicitely via the journal, but a large number of inodes can be dirtied without filling the journal resulting in a very long sync time when the sync finally does happen. The flusher was not an adequate solution to this problem since it forced writeback of small writes too eagerly killing performance.

WBThrottle tracks unflushed io per hobject_t and ::fsyncs in lru order once the start_flusher threshhold is exceeded for any of dirty bytes, dirty ios, or dirty inodes. While any of these exceed the hard_limit, we block on throttle() in _do_op.

See src/os/WBThrottle.h, src/osd/WBThrottle.cc

To track the open FDs through the writeback process, there is now an fdcache to cache open fds. lfn_open now returns a cached FDRef which implicitely closes the fd once all references have expired.

Filestore syncs have a sideeffect of flushing all outstanding objects in the wbthrottle.

lfn_unlink clears the cached FDRef and wbthrottle entries for the unlinked object when then last link is removed and asserts that all outstanding FDRefs for that object are dead.