LAST_EPOCH_STARTED

info.last_epoch_started records an activation epoch e for interval i such that all writes commited in i or earlier are reflected in the local info/log and no writes after i are reflected in the local info/log. Since no committed write is ever divergent, even if we get an authoritative log/info with an older info.last_epoch_started, we can leave our info.last_epoch_started alone since no writes could have commited in any intervening interval (See PG::proc_master_log).

info.history.last_epoch_started records a lower bound on the most recent interval in which the pg as a whole went active and accepted writes. On a particular osd, it is also an upper bound on the activation epoch of intervals in which writes in the local pg log occurred (we update it before accepting writes). Because all committed writes are committed by all acting set osds, any non-divergent writes ensure that history.last_epoch_started was recorded by all acting set members in the interval. Once peering has queried one osd from each interval back to some seen history.last_epoch_started, it follows that no interval after the max history.last_epoch_started can have reported writes as committed (since we record it before recording client writes in an interval). Thus, the minimum last_update across all infos with info.last_epoch_started >= MAX(history.last_epoch_started) must be an upper bound on writes reported as committed to the client.

We update info.last_epoch_started with the intial activation message, but we only update history.last_epoch_started after the new info.last_epoch_started is persisted (possibly along with the first write). This ensures that we do not require an osd with the most recent info.last_epoch_started until all acting set osds have recorded it.

In find_best_info, we do include info.last_epoch_started values when calculating the max_last_epoch_started_found because we want to avoid designating a log entry divergent which in a prior interval would have been non-divergent since it might have been used to serve a read. In activate(), we use the peer's last_epoch_started value as a bound on how far back divergent log entries can be found.

However, in a case like

```
calc_acting osd.0 1.4e( v 473'302 (292'200,473'302] local-les=473 n=4 ec=5 les/c 473/473 556/
calc_acting osd.1 1.4e( v 473'302 (293'202,473'302] lb 0//0//-1 local-les=477 n=0 ec=5 les/c 
calc_acting osd.4 1.4e( v 473'302 (120'121,473'302] local-les=473 n=4 ec=5 les/c 473/473 556/
calc_acting osd.5 1.4e( empty local-les=0 n=0 ec=5 les/c 473/473 556/556/556
```

since osd.1 is the only one which recorded info.les=477 while 4,0 which were the acting set in that interval did not (4 restarted and 0 did not get the message in time) the pg is marked incomplete when either 4 or 0 would have been valid choices. To avoid this, we do not consider info.les for incomplete peers when calculating min_last_epoch_started_found. It would not have been in the acting set, so we must have another osd from that interval anyway (if maybe_went_rw). If that osd does not remember that info.les, then we cannot have served reads.