Problem Statement:

“Opening “pristine” region without waiting for Log splitting/replaying to complete
“

When a region server dies, we process its logs (log splitting/replaying), and then assign regions that were hosted on the dead regionserver. There may be some “pristine” regions, which could be open without log replaying to complete. We should be able to open such regions without delay. Some potential use cases can be read intensive workload, multi table regions deployed on a regionserver and writes are being done only on few tables, etc.

By pristine, it means those regions whose edits are flushed and they don’t have any dependency on WAL.

Scope: The scope of this doc is to handle a regionserver shutdown.
Handling an entire cluster shutdown/master-shutdown is beyond its scope.

Proposed Solution:

Use HServerLoad to help master decide whether a region should wait for WAL replay or not.

Preconditions:
1) .META. is available, and
2) Master is in steady state all time throughout the process.

With separate meta log, we meet the first condition. In case Master goes down during the process, we fallback to the existing mechanism.

A regionserver sends a status report to the master as part of a heartbeat. Its frequency is set by “hbase.regionserver.msginterval” (default 3 sec).
The heartbeat is composed of requestsLoad, heap-usage, and region level information of all the deployed regions on that server. Below is a sample of serverload.

Sample ServerLoad:
numberOfRequests: 0 totalNumberOfRequests: 732280 usedHeapMB: 856 maxHeapMB: 4083 ***regionLoads*** *{ regionSpecifier { type: REGION_NAME value: ".META.,,1" } stores: 1 storefiles: 1 storeUncompressedSizeMB: 0 storefileSizeMB: 0 memstoreSizeMB: 0 storefileIndexSizeMB: 0 readRequestsCount: 1548 writeRequestsCount: 3 totalCompactingKVs: 117 currentCompactedKVs: 93 rootIndexSizeKB: 0 totalStaticIndexSizeKB: 0 totalStaticBloomSizeKB: 0* ***completeSequenceId****: **940** }...*

We add a boolean variable in the ServerLoad in the *regionLoads* section of the request: **allWALEntriesFlushed.** This is added for all regions across the regionserver.
This variable is used to identify whether a region has all its mutations flushed to HFiles or not. If it is true, that means this region has no dependency on WAL and can be open right away by the master in the event of processing a regionserver shutdown event (in the ServerShutdownHandler class). Whenever a region is

opened/flushed, its allWALEntriesFlushed is set to true. When it receives the first mutation after a flush/open, it is set to false.

HMaster: It processes and extracts some useful information for the ServerLoad. It stores completeSequenceId for all the regions. A completeSequenceId is the last flushed SequenceId, i.e., any WAL entry for that region with WALEdit#key#sequenceId lesser than above reported sequenceId is already flushed; therefore, it can be ignored while doing WAL replay.
In HMaster, we add a map for region:allWalEntriesFlushed, and also a max of completeSequenceId (max_completeSequenceId) across all the regions for a given regionserver. This will be used when processing the regionserver shutdown event. The ServerShutdownHandler thread reads the log directory of the dead regionserver.
When processing the regionserver shutdown event, the ServerShutdownHandler reads the region:allWalEntriesFlushed map. It then starts reading the WAL files for that regionserver, starting with the most recent WAL file. It reads it till the end and update allWalEntriesFlushed mapping of regions for WALEdit entries with sequenceId greater than the recorded max_completeSequenceId. This is to cover all the regions that were updated after the last ServerLoad report and shutdown event. Once it has read the WAL files (usually only the last one) which have sequenceIds greater than max_completeSequenceId, it sends an open request for regions which has allWALEntriesFlushed set to true, as they don't need to wait for log splitting/replaying to complete. In order to avoid reading more than one WAL file in this process, we will send a regionserver report on every log rolling event to update the region:allWalEntriesFlushed and max_completeSequenceId.

This will make pristine regions available without waiting for log split. This process can be done in parallel with existing log splitting/replaying.

Pros:
1) Simple. It uses the existing server load request with no additional cost (other than adding a dirty bit for a region). With protobufs, it is fairly simple to do this.

Con:
1) Double reading of last WAL: There may be a case when the last WAL might be read twice: one by this process and another by a LogSplitWorker. We might add an optimization later on to simultaneously process the last WAL and not to add it in the log split task queue.