Architecture for HydraBase

- **Overview**
  - Today, at Facebook, HBase is used in production by many Facebook services with different types of workloads. This includes our internal monitoring system(ODS), the recently launched Nearby Friends feature, search indexing, streaming data analysis, and data scraping for our internal data warehouses and the biggest of all Facebook Messages. As we scale these systems to meet the ever increasing growth reliability becomes a very important factor. Over the past couple of years we have continued to make incremental improvements to HBase to improve the overall reliability. With this effort we are able to achieve 4 9's of availability in steady state. Among the challenges that we encounter today with HBase availability are the time to recover individual failing hosts, rack-level failures, and the nuances of running on dense storage. One way of addressing these availability issues is to run HBase in a master-slave setup, asynchronously replicating updates between the two clusters. But this requires failover to be carried out at the cluster level, causes outages on the order of minutes of downtime when failing over from master to slave, and potentially results in data loss due to the asynchronous nature of the replication.
  - Hence, in order to improve the overall availability of the HBase, we need take a step back and look at the overall architecture of HBase to make the required improvements. This document describes the overall requirements and the architectural changes proposed.

- **Requirements**
  - No single point of failure in the system. Handle Single disk/Single Host/Rack Level/Cluster level failures.
  - No data loss at cluster level failures
  - Per Shard failover capability
  - All failures should be quick to recover from (order of seconds) - otherwise the user will see the failure/unavailability.
  - No cascading failures typical to a distributed system. In other words, small groups of nodes should communicate only amongst themselves.
  - To tolerate single DC issues transparently, the solutions needs to do some form of synchronous replication across datacenters. But, in order to meet the

availability guarantees in the face of single DC failures, we need some form of distributed consensus (for example, paxos). While this might have an impact on write latency, it should not affect the write throughput.

- From an operations/software upgrade point of view, the ability to roll-out new changes in a controlled fashion, without the necessity to roll it out over the entire set.
- Load balancing is removed out of the reliability path. Also, placement choices are made far less frequently.
- LSM (log structured merge trees) based storage model for write heavy workloads.

- **Overall changes**
  - There won't be any changes related to the concept of Tables as in HBase. As in HBase, each table will be divided into regions and the RegionServers will be the hosting the regions.
  - In order to tolerate multiple failures domains and have the ability to recover within a few seconds, each region will be hosted by a set of Region Servers, which we call as the replicas. This set will be stored into a globally available location called as the RMAP. The replicas can be distributed in/across racks and can span across multiple clusters. This set of replicas are collectively responsible for serving the region and making sure that its available as long as there is quorum.

  - **Replication Protocol**
    - At any given point in time there will be only one leader amongst the set, who will be serving all the read and write requests to the client. The election of the leader will be done using the RAFT protocol. Having only one peer serving all the requests provides the required strong consistency which HBase currently guarantees.
    - Each replica will have its own Write Ahead Log(WAL) which will be stored on local storage media, unlike HBASE, where its stored in HDFS.
    - All the writes will be replicated synchronously by the leader to the replica set. All the reads will be served locally by the leader.

- **RMAP**
  - RMap contains the quorum configuration information for each Region. Typically, each quorum has F replicas across different DCs. Based on the network latency to the client, each DC will have a rank number. The DC, which has the lowest network latency to the client, will have the highest rank. This DC-level rank information indicates the preferred leader location for the quorum. For example, if any qualified quorum member has a higher DC ranking is able to take over the leadership, the current active leader in the low-rank DC will step down voluntarily.
  - In addition, each quorum member has a machine-rank in order to speed up the bootstrapping. The replica, which has higher rank (DC-rank + machine-rank), will have a lower election timeout. Therefore, the highest rank machine will be first one to propose for the leadership and most likely that machine will be the first leader. Furthermore, the quorum members in the highest DC rank will have much lower election timeout than those in other DCs. Therefore, unless all the quorum members in the highest rank DC are failed, the leadership won't shift to the replicas in the lower ranking DC.

- **Types of replicas**
  - Each replica will have an associated role with it. The different types of roles are:
  - **ACTIVE**: An active replica is the one who is performing all the LSM operations in the RegionServers. This includes flushes and compactions. Since, we have one HDFS cluster per physical cluster, there will be only ONE Active replica per HDFS cluster. This ensures the consistency with to the mutation of data done in HDFS. By default, the current leader is always ACTIVE.
  - **ACTIVE-WITNESS**: An active-witness replica is the one who has a memstore associated with it, but is not performing any LSM operations that mutate the data the HDFS. These can be considered as read-replicas, which apply all the mutations being replicated by the leader to the memstore. There can be one or more active-witness replicas per HDFS cluster.
  - **SHADOW-WITNESS**: A shadow-witness replica is the one who is only participating in the replication via the protocol. Is does not perform any mutating operations to the HDFS cluster.

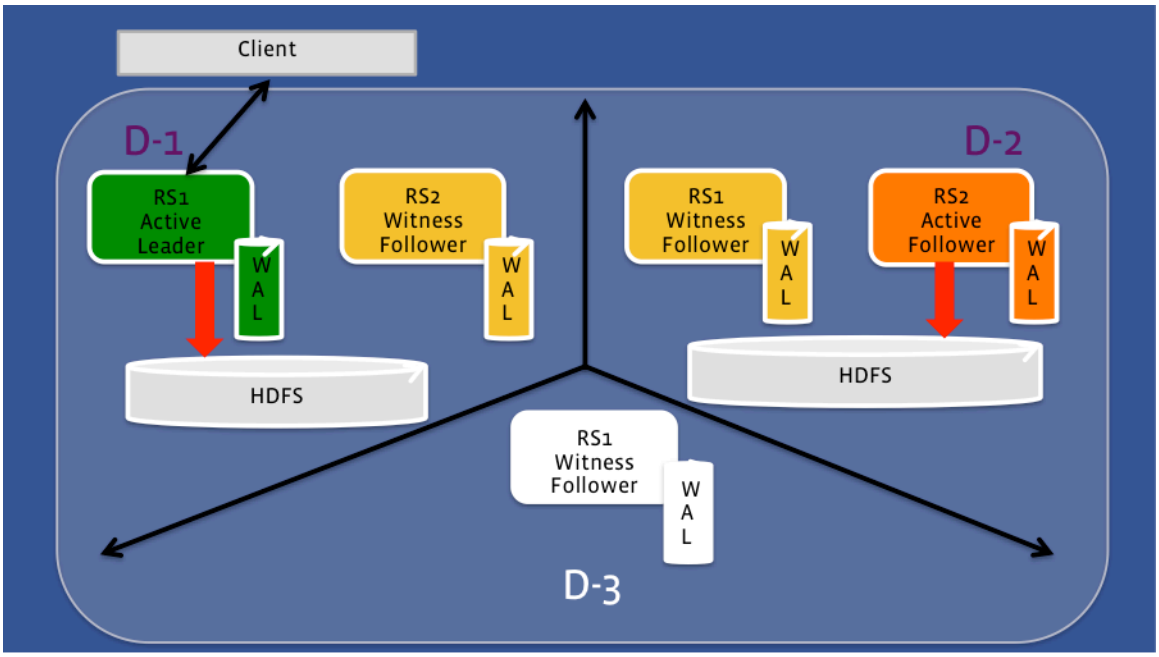
- **Distributed Storage Layer**
  - Currently the common deployments of HBase are within the same cluster using only one HDFS cluster. Because of this deployment, HBase cannot tolerate cluster level failures without losing data consistency. With async replication support in HBase, it can do cluster level failovers but there can be data loss in case of unplanned failovers.
  - In HydraBase, one cluster can span multiple physical clusters. There will be one HDFS cluster per each physical cluster will be used as the Distributed Storage Layer for that cluster.

- **Failure Scenarios Analysis**
  - Following table list the various failure across multiple domains and how does HydraBase handle these failures.

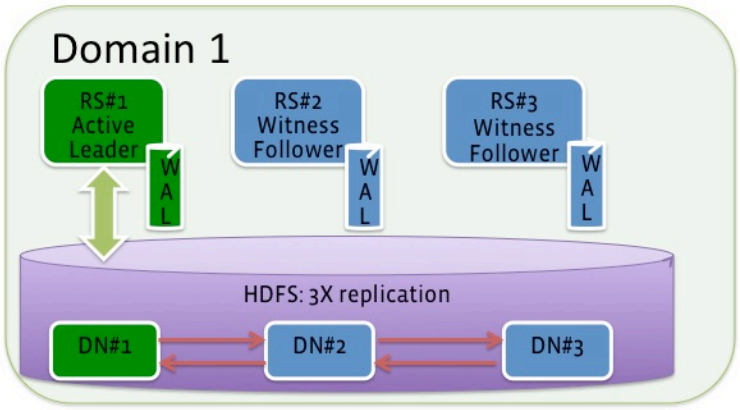
	A	B
1	**Failure Scenario**	**Expected Quorum Behavior**
2	**Primary Active Leader Failure**	The primary active-witness replica will become the next leader by design. Because the primary active-witness replica, which has a lower timeout than others, will be first one to propose a larger term. After the active witness replica becomes the leader, it will promote itself to be the active replica and start to flush and compaction. When the original primary active replica came back, the current leader will downgrade this replica into a witness replica. Nothing changes in the secondary DC and tertiary DC.
3	**Secondary Active Replica Failure**	The leader will detect the failure from the heartbeat RPC and promote the secondary witness replica as active replica.
4	**Up to 2 Witness Replicas Failures**	The leader will do nothing but wait for either the machine coming back online or load balancer to update the quorum.
5	**Primary Active Replica + Primary Witness Replica Failures (Primary DC is down)**	The active replica in the secondary DC will be the leader and continue to serve requests. In the meanwhile, it will wai for either the machines coming back online or load balancer to update the quorum.
6	**Secondary Active Replica + Secondary Witness Replica Failures (Secondary DC is down)**	The leader will do nothing but wait for either the machines come back online or load balancer to update the quorum.

	A	B
7	**Primary Active Replica + Secondary Active Replica Failures**	The witness replica in the primary DC will be elected as new leader and promote itself as active replica. In the meanwhile, it will promote another witness replica in the secondary DC to be the active replica. After that, the scenario will be equivalent to multiple witness replica failure case.

- **Deployment Scenarios**
  - **Multi Cluster Deployment Setup**
    - The following is a multi cluster deployment for HydraBase with replica set size of 5 across 3 different physical clusters. D-1 and D-2 have one physical HDFS cluster and 2 replicas each. The client is closer to D-1 and the current leader is RegionServer-1(RS-1) in D-1.
    - RS-1 is the ACTIVE in D-1 and will be performing all the LSM operations and also serving all the client requests. RS2 in D-1 will be an ACTIVE-WITNESS. While RS2 in D-2 will be the ACTIVE replica performing the flushes/compactions to the HDFS cluster in D-2.
    - Each of the replicas has its own WAL stored on local storage and not in HDFS.
    - D-3 does not have HDFS, hence it cannot serve client requests. However, in-order to guarantee correctness it can still become the leader. This can happen in the scenario where all the available replicas are lagging and in-order to guarantee data consistency, RS-1 in D-3 has to become the leader. In case it becomes the leader, the leadership will failover to the other replicas (with the backing HDFS) as soon as the replicas are caught-up. so that it can start serving client traffic. In order to minimize this scenario, the rank for the replica in D-3 will be the lowest, '1' in this case. Because of this, it will have the highest progress timeout and least probability of becoming the leader.

- 

    - **Single(In) Cluster Deployment setup**
        - HydraBase can be deployed in an in-cluster mode also. Below represents a deployment scheme for in-cluster setup with replica set of size 3 and one HDFS cluster. There will be only one ACTIVE replica and the others can be ACTIVE-WITNESS/WITNESS replicas.



    -