Opening remarks:

Chris Kelly, open source program Salesforce, introduction.

- Helping do more open source at Salesforce, all aspects.
    - Licensing
    - Tooling
    - Insight

Lars Hofhansl: Phoenix at Salesforce

- Lots of use cases, different types of workloads and different business use-cases
- Salesforce doesn't fork!
- Phoenix: ~100 clusters, 17B requests/day, 11PB of data (~2PB before replication)
- HBase opentsdb ~15T events
- Hadoop 50PB cluster

Introductions of attendees

Discussion Points

- Too many branches!! Hard to maintain so many.
    - Do we need 1.2, 1.3 and 1.4 branches? Can we get away with one?
- Salesforce keeps Phoenix close to open-source release
- Cross-release compatibility -- rolling-upgrade, steps to do this?
- Give more confidence around minor-release upgrades
- Better clarity around "boundaries" for clients
    - Did catalog schema change?
    - Did encoding of data change?
- Too many hbase 1.x branches?
- Who are the major deployers of Phoenix?
    - *poll dev@ and user@ to figure out who uses what HBase versions*
    - Can we move away older 1.x versions of HBase?
    - Can we push HBase 2.x to move to newer versions and avoid the same 1.x branch problems in HBase?
- ***Need to make more committers!!***
- *Steal HBase's FindFlakeyTests jenkins logic*
-
- What is the direction that Phoenix needs to take?
    - Split up Phoenix into 4 discrete pieces: types, executor, RS-side, …
        - How do we proceed with splitting this up: high-level separation of responsibilities and logic.

        - Phoenix needs to drive composition of the discrete pieces into a complete system
    - Build a client-only phoenix if we have a type system
- Cloud, non on-prem
    - Ratis LogService to replace WALs
    - De-couple from POSIX filesystem guarantees
        - S3a with dynamodb is not sufficient
    - Big goal is what? Drive adoption of hbase/phoenix? Cost-savings?
- Phoenix today is two level query executor: client and RS
    - Does a more complex model in Phoenix help?
    - Should we rely on Presto, Spark, Hive, other?
- Centralized schema management is nice to have.
    - Is Hive metastore the right way to do it?
- Hidden issues around multi-tenant queries: PHOENIX-4657
    - Problems when deleting a row from a multi-tenant table may cascade to a different tenant's index/view
- “Project cleanliness”
    - UT/IT do not consistently pass
    - HBaseMiniCluster “force-stop” to skip all close region logic, greatly shorten test execution time
    - Can we get a sponsor for better hardware for running UT/IT on ASF infra?
- HBase incompatibilities
    - Method additions in HBase are hard in Phoenix. Do JDK8 default methods help? What is the current pain in hbase2?
- Lots of new devs to HBase/Phoenix
    - How can we engage lots of new engineers to the project?
    - How can we ensure we review patches in a timely manner?
- How can we get more reviewer bandwidth?
    - Have recommendations of what a patch describes what it does
        - Comments in the code
        - Formatted code
        - In review board
    - Phoenix code quality makes patches hard:
        - touch many lines through out files
        - unit tests aren't compartmentalized (or list preconditions)
        - pre-checkin hook to require unit test coverage increase
        - style checker
    - Improve flaky test noise Steal HBase's FindFlakeyTests jenkins logic
- Metadata ops are multi-stage and not necessarily idempotent
    - Procv2 is meant to help some of this, but phoenix is doing things to fast to use pv2 effectively.
    - PV2 is just too heavy for what Phoenix needs?

    - Can we use state machines to better define what phoenix operations, better understand
    - Can we write a better API to help us do better
- Index rant by LarsH
    - Global and local, mutable and immutable
    - Transactional indexes are possible, but not here yet
    - Percolator style for mutable and immutable indexes, driven client side
    - Make local indexes solid
    - Transactional local indexes via Omidv2