Potential HBase Modules

- util
  - generic utilities non-specific to hbase
- common
  - central classes and interfaces seen by all downstream modules
  - covered by fast unit tests
  - hbase specific data structures like KeyValue, KeyValueComparator, KeyValueScanner, KeyValueHeap with tests
  - KeyValue Filters with tests
  - interfaces for client to communicate with region, regionserver, master
  - DataBlockEncoder interface
  - Add interfaces for other HFileBlock BlockTypes
- codec, prefix-trie, and other codec contribs
  - implementations and unit tests for HFileBlock interfaces defined in hbase-common
  - Data
    - DataV1
    - DataV2, ChecksummedData
    - EncodedData
      - PrefixData, DiffData, FastDiffData, PrefixTrieData
  - Index
    - IndexV1
    - RootIndex, IntermediateIndex, LeafIndex
    - (future PrefixTrieIndex encodings)
  - Bloom
    - BloomChunk, GeneralBloomMeta, DeleteFamilyBloomMeta
  - Meta
    - FileInfo, Trailer
- memstore
  - in-memory data structure
  - simple interface, but complex implementation
  - optimize for performance, compactness, and concurrency
  - thorough unit/performance tests
  - isolate to allow for pluggable implementations (PrefixTrieMemstore)
- wal
  - like memstore, simple interface with possibly complex implementation
  - coordinate multiple log files
  - auto-optimize log selection and group-commit timing
  - support cluster replication?
  - support durable async secondary indexes?
- region
  - isolated code and tests for a single region
  - HFile, StoreFile, HalfStoreFile, Store

    - intra-row transactions, RWCC
    - locality groups
    - KeyValue cache
    - optimized Filter execution
    - flush/compaction execution and optimization
    - coprocessor hook implementation
    - metrics collection
    - performance regression tests
    - thorough data/transaction integrity testing
- server
    - hold a bunch of open regions
    - block cache
    - auto-tune memstore and block cache sizes
    - schedule flushes/compactions
    - split/merge regions
    - expose api's to clients and throttle/prioritize requests
        - distributes/aggregates client requests between local regions
    - catch exceptions, retry failures,
    - initiate and participate in cross-region transactions
    - aggregate and expose region metrics
- master
    - use client api to read ROOT and META tables?
    - assign regions to servers
    - coordinate log replay after failures
    - aggregate metrics from regionservers
- integration-test
    - catch-all for tests including multiple servers
    - minicluster tests for region placement, regionserver failures, retries, repairs, cluster performance, replication, splits, merges, stuck regions, transactions, map/reduce interfaces, distributed log replay, online schema change, etc