Flaky tests and how to avoid them

Here is a list of some flaky tests that cause build failure. Most of them can be avoided if we pay more attention when writing tests.

Issue: Hit the default 64 connection max limit and next connection attempt blocks and builds are hanging.

- **Fix:** When restart service, use check_call() instead of call(), and use the close_fds argument to make sure the TCP socket is closed in the spawned child process.
- IMPALA-2605 Builds are hanging in mini stress test
- **Longer term fix**: Use fixtures and service wrappers so that test writer does not have to manage the connection/subprocess state

Issue: Some supported file systems like S3 and Isilon build don't have Hive deployed in the test setup

- **Fix:** if your test uses hive to run any query, you should mark it with "@SkipIfS3.hive" and "@SkipIfIsilon.hive"
- IMPALA-2504
- For a complete list of what not implemented or supported, check skip.py

- **Longer term fix**: Switch to tagging tests with required config options and test areas, e.g. 'hive', 'isilon', that can be standard, and do not run tests that are not appropriate for a configuration automatically. The test writer should have to know what standard config options are important for the test, but not all the other config options that may require the test to be skipped.

Issue: Impala supports multiple FileSystem now, HDFS, S3, etc. Hardcode filesystem prefix will cause test failure on another filesystem.

- **Fix:** use get_fs_path() to prepend the correct filesystem prefix
- IMPALA-2488

Issue: Test failed due to read-your-writes metadata failure. error "AnalysisException: Table does not exist: ...". This is because we try to run EE test in parallel. but some EE tests perform concurrent invalidate metadata and DDL operations. This combination of operations may lead to unexpected outcomes for the DDL operation, e.g., a create table succeeded, but the table appears to have not been added, and similar issues.

- **Fix:** Avoid calling "invalidate metadata" when possible. if it's really needed, mark the test with "@pytest.mark.execute_serially" to avoid this concurrency issue.
- IMPALA-2687

Issue: Test failed with error "AnalysisException: Table already exists:". We run the same test with different query options in parallel. If your tests use temp table and does "DROP TABLE" then "CREATE TABLE", one instance could already create the table while another instance tries to create it.

- **Fix:** Randomizing table names so the multiple instances will use different table name and won't have this race issue.
- IMPALA-2460
- IMPALA-2537

Issue: Missing or mismatch metric "AssertionError: Metric value external-data-source.class-cache.misses did not reach value"

- **Fix:** ImpalaCluster.get_first_impalad() may NOT return the same impalad as that which the test class's self.client was connected to. Uses the ImpaladService created on the base class instead to ensure it's the same impalad as the impalad that the client is connected to.

- IMPALA-2220

Issue: Test fail because it cannot retrieve query profile from Impalad

- **Fix:** Make sure you send query and retrieve query status from the same Impalad
- IMPALA-2103

Issue: Our test loading usually do compute stats for tables but not all. if your test rely on a table has stats computed, it might fail

- **Fix:** using a table that guarantee have stats computed, or modify your tests to not rely on stats computed.

- IMPALA-2801

- **Todo:** List of tables that we purposely do not compute stats. e.g. functional_hbase.alltypes

Issue: QueryTestResults mismatch. Impala is a distributed query engine, multiple impalads process data in parallel and try to return result asap. There is no guarantee which Impalad returns result first. the result could vary from run to run. e.g. query with LIMIT 10 could return different rows each time.

- **Fix:** If your test checks returned resultset, make sure they are ordered (Add ORDER BY) and the resultset is deterministic

- IMPALA-2497

Issue: Planner test failure due to file size vary from run to run. One cause is Parquet file size could change when impala version change because Impala put the version string is in file header.

- **Fix:** In most of tests, file size doesn't matter, your test should ignore file size.
- IMPALA-2565

Issue: Test failed due to "Query referencing nested types is not supported because the --enable_partitioned_hash_join and/or --enable_partitioned_aggregation Impala Daemon start-up flags are set to false."

- **Fix:** If possible, move tests into a different .test file which is already skipped when using the legacy aggs/joins (@SkipIfOldAggsJoins. nested_types in pytest files). Alternatively, add the marker to those pytests.
- IMPALA-2894

Issue: Your python test creates a table to play in, but a `use` command from a previous test causes a change in which database the table is created, or a separate concurrent test deletes the database or table underneath the original test. Or, you have tests that have to run serially because they do INSERTs or DROPs or CREATEs.

- **Fix:** Use the `unique_database` `impala-py.test` fixture. This creates a database with a unique name based on the test ID checksum. This means the current test (even specific to vector) will be the only test to use that database. Run `impala-py.test --fixtures` (from `tests/`) for more info.
- IMPALA-3195, review 2586

Issue: Test works on your development platform, but not on RHEL5 or RHEL6 or some other OS

A few other things to consider:

- .test format could change. We keep adding more stuff to table stats, so "SHOW TABLE STATS" result could change from release to release. For example, We add "INCREMENTAL STATS" and "LOCATION". When backport fixes to earlier release, need to pay attention if this changes and revise the .test accordingly.
- "mem_limit_exceeded" in concurrent test env, one test could be affected by other concurrent tests.

- Impala running on EC2 could be slower, and response took longer IMPALA-2621
- In general, always use fully qualified names for the specified tables to avoid confusion. for example: functional_parquet.alltypes