COMMUNITY
THE ASF CONFERENCE
CODE

Property / Fuzz Testing In Apache Cassandra

David Capwell, Software Engineer at Apple, Committer for Apache Cassandra

Agenda

Sorry if this is fuzzy...

- Why should you care?
- Anatomy of a Property Test
- What is a Stateful Property
- Role of Bias
- What is a Simulation Property

Why should you care?

How many edge cases really exist?

- As developers we know all edge cases that could exist, right?
- How one change will interact with all other systems? Right?
- My change is simple!

Why should you care?

How many edge cases really exist?

```
createTable("CREATE TABLE %s(pk int, v int, PRIMARY KEY(pk))");
execute("INSERT INTO %s(pk, v) VALUES (?, ?)", 0, 0);
assertRows(execute("SELECT * FROM %s WHERE pk=?", 0),
           row(0, 0));
```

Why should you care?

How many edge cases really exist?

```
createTable("CREATE TABLE %s(pk int, ck int, v int, PRIMARY KEY(pk, ck))");
execute("INSERT INTO %s(pk, ck, v) VALUES (?, ?, ?)", 0, 0, 0);
assertRows(execute("SELECT * FROM %s WHERE pk=?", 0),
           row(0, 0, 0));
```

Why should you care?

How many edge cases really exist?

```
createTable("CREATE TABLE %s(pk int, ck int, v int, PRIMARY KEY(pk, ck)) WITH CLUSTERING ORDER BY (ck DESC)");
execute("INSERT INTO %s(pk, ck, v) VALUES (?, ?, ?)", 0, 0, 0);
assertRows(execute("SELECT * FROM %s WHERE pk=?", 0),
           row(0, 0, 0));
```

Why should you care?

How many edge cases really exist?

```
createTable("CREATE TABLE %s(pk text, ck text, v text, PRIMARY KEY(pk, ck)) WITH CLUSTERING ORDER BY (ck DESC)");
execute("INSERT INTO %s(pk, ck, v) VALUES (?, ?, ?)", "a", "b", "c");
assertRows(execute("SELECT * FROM %s WHERE pk=?", "a"),
           row("a", "b", "c"));
```

Why should you care?

How many edge cases really exist?

- We covered every case right? We know this is working…
- …
- Its working?

Why should you care?

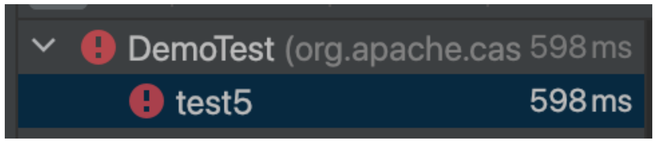
How many edge cases really exist?

```
schemaChange(format("CREATE TYPE %s.test5 (a int)", KEYSPACE));
String name = createTableName(null);
String fq = format("%s.%s", KEYSPACE, name);
createTable(format("CREATE TABLE %s(pk int, ck frozen<%s.test5>, v int, PRIMARY KEY(pk, ck)) WITH CLUSTERING ORDER
BY (ck DESC)", fq, KEYSPACE));
execute(format("INSERT INTO %s(pk, ck, v) VALUES (?, ?, ?)", fq), 0, tuple(0), 0);
assertRows(execute(format("SELECT * FROM %s WHERE pk=?", fq), 0),
           row(0, tuple(0), 0));
```

Why should you care?

How many edge cases really exist?

```
schemaChange(format("CREATE TYPE %s.test5 (a int)", KEYSPACE));
String name = createTableName(null);
String fq = format("%s.%s", KEYSPACE, name);
createTable(format("CREATE TABLE %s(pk int, ck frozen<%s.test5>, v int, PRIMARY KEY(pk, ck)) WITH CLUSTERING ORDER
BY (ck DESC)", fq, KEYSPACE));
execute(format("INSERT INTO %s(pk, ck, v) VALUES (?, {a: 0}, ?)", fq), 0, 0);
assertRows(execute(format("SELECT * FROM %s WHERE pk=?", fq), 0),
           row(0, tuple(0), 0));
```




```
org.apache.cassandra.exceptions.InvalidRequestException: Invalid user type literal for ck of type frozen<test5>
    at org.apache.cassandra.cql3.terms.UserTypes$Literal.validateAssignableTo(UserTypes.java:175)
    at org.apache.cassandra.cql3.terms.UserTypes$Literal.prepare(UserTypes.java:137)
    at org.apache.cassandra.cql3.terms.Terms$Raw$2.prepare(Terms.java:280)
```

See https://issues.apache.org/jira/browse/CASSANDRA-19781

Why should you care?

How many edge cases really exist?

- Cleary that was the last edge case right?
- …
- Right?

Why should you care?

How many edge cases really exist?

```
BigInteger pk = BigInteger.valueOf(-42 + -42);
BigInteger ck = BigInteger.valueOf(42 + 42);
BigInteger v = BigInteger.valueOf(-8200 + -16990);
createTable("CREATE TABLE %s(pk varint, ck varint, v varint, PRIMARY KEY(pk, ck))");
execute("INSERT INTO %s(pk, ck, v) VALUES (-42 + -42, 42 + 42, -8200 + -16990)");
assertRows(execute("SELECT * FROM %s"),
           row(pk, ck, v));

assertRows(execute("SELECT * FROM %s WHERE pk=?", pk),
           row(pk, ck, v));
```

```
java.lang.AssertionError: Got less rows than expected. Expected 1 but got 0
    at org.junit.Assert.fail(Assert.java:88)
    at org.junit.Assert.assertTrue(Assert.java:41)
    at org.apache.cassandra.cql3.CQLTester.assertRows(CQLTester.java:2018)
```

See https://issues.apache.org/jira/browse/CASSANDRA-19874

Why should you care?

How many edge cases really exist?

```
BigInteger pk = BigInteger.valueOf(59463118).multiply(BigInteger.valueOf(-2171));
BigInteger ck = pk;
BigInteger v = pk;
createTable("CREATE TABLE %s(pk varint, ck varint, v varint, PRIMARY KEY(pk, ck))");
execute("INSERT INTO %s(pk, ck, v) VALUES (59463118 * -2171, 59463118 * -2171, 59463118 * -2171)");
assertRows(execute("SELECT * FROM %s"),
           row(pk, ck, v));
```

```
java.lang.AssertionError: Invalid value for row 0 column 0 (pk of type varint), expected <-129094429178> but got
<-245410298>
Invalid value for row 0 column 1 (ck of type varint), expected <-129094429178> but got <-245410298>
Invalid value for row 0 column 2 (v of type varint), expected <-129094429178> but got <-245410298>
```

See https://issues.apache.org/jira/browse/CASSANDRA-19874

How can we do better?

- Accept that we don’t know every edge case
- But, we can generate them…
- So… how do we do that?

Property Testing

Anatomy of a Property Test

- Source of Randomness
- 1 or more Properties
- Multiple examples
- Reproducible and Replayable
- Generate Data

Anatomy of a Property Test

Source of Randomness

```
interface RandomSource {
  long nextLong(long min, long max);
}
```

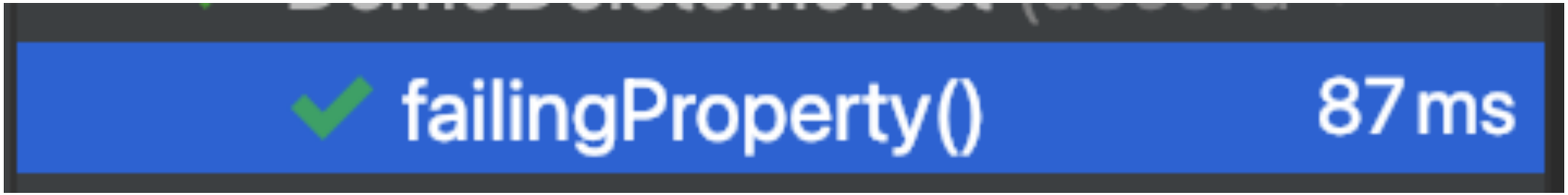
Anatomy of a Property Test

A Property

```
RandomSource rs = new DefaultRandom();
long input = rs.nextLong(0, 100);
// property
assertThat(input/input).isEqualTo(1);
```

Anatomy of a Property Test

A Property



Anatomy of a Property Test

Look Again

```
RandomSource rs = new DefaultRandom();
long input = rs.nextLong(0, 100);
// property
assertThat(input/input).isEqualTo(1);
```

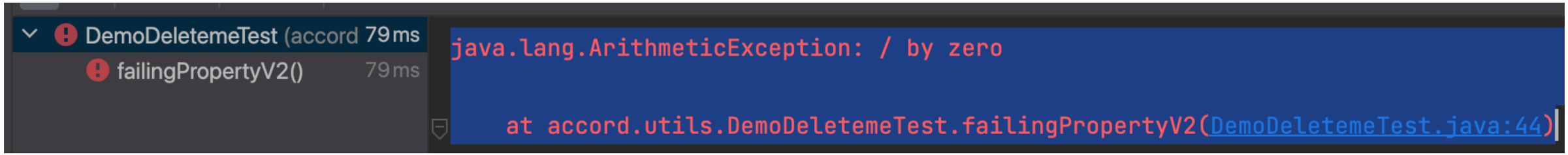
Anatomy of a Property Test

Multiple Examples

```
RandomSource rs = new DefaultRandom();
for (int i = 0; i < 1000; i++) {
  long input = rs.nextLong(0, 100);
  // property
  assertThat(input/input).isEqualTo(1);
}
```

Anatomy of a Property Test

Multiple Examples



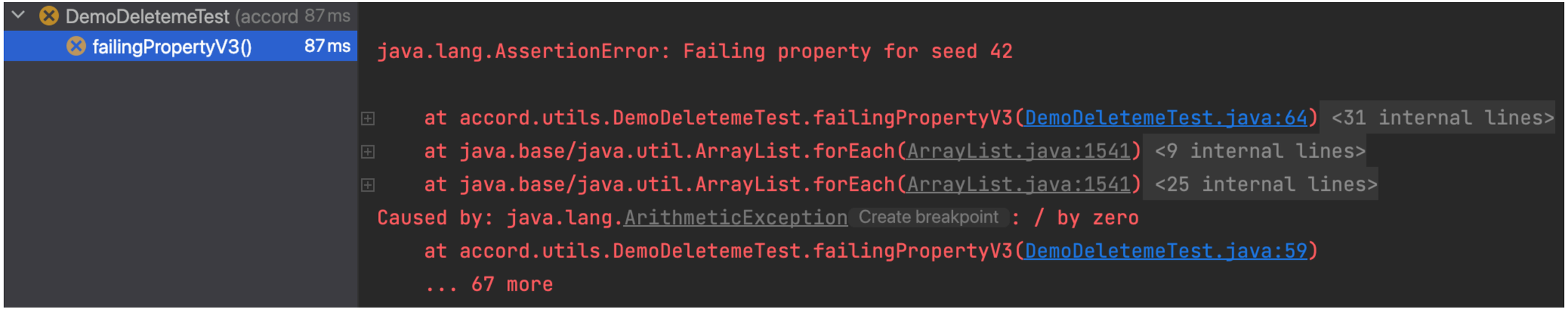
Anatomy of a Property Test

Reproducible and Replayable

```
long seed = 42;
RandomSource rs = new DefaultRandom(seed);
try {
  for (int i = 0; i < 1000; i++) {
    long input = rs.nextLong(0, 100);
    // property
    assertThat(input/input).isEqualTo(1);
  }
} catch (Throwable t) {
  throw new AssertionError("Failing property for seed " + seed, t);
}
```

Anatomy of a Property Test

Reproducible and Replayable



Anatomy of a Property Test

Generate Data

```
Gen<UUID> randomUUID = rs -> {
  long most = rs.nextLong();
  most &= 0x0f << 8; /* clear version        */
  most += 0x40 << 8; /* set to version 4     */
  long least = rs.nextLong();
  least &= 0x3fl << 56; /* clear variant        */
  least |= 0x80l << 56; /* set to IETF variant  */
  return new UUID(most, least);
};
```

```
Gen<InetAddress> ipv4 = rs -> {
  byte[] bytes = new byte[4];
  ByteArrayUtil.putInt(bytes, 0, rs.nextInt());
  return InetAddress.getByAddress(bytes);
};
```

```
enum Status { Pending, Running, ShuttingDown, Down }
Gen<Status> statusGen = rs -> {
  return Status.values()[rs.nextInt(0, Status.values().length)];
};
```

Anatomy of a Property Test

Generate Data: Complex Types

```
Gen<KeyspaceMetadata> ksGen = rs -> {
  String name = nameGen.next(rs);
  KeyspaceMetadata.Kind kind = kindGen.next(rs);
  ReplicationParams replicationParams = replicationGen.next(rs).withKeyspace(nameGen).build().next(rs);
  boolean durableWrites = durableWritesGen.next(rs);
  KeyspaceParams params = new KeyspaceParams(durableWrites, replicationParams);
  Tables tables = tablesGen.next(rs);
  Views views = viewsGen.next(rs);
  Types types = typesGen.next(rs);
  UserFunctions userFunctions = userFuncGen.next(rs);
  return KeyspaceMetadata.createUnsafe(name, kind, params, tables, views, types, userFunctions);
};
```

Anatomy of a Property Test

Generate Data

- Hundreds of non-Cassandra generators exist today!
- Need an identifier? Generators#IDENTIFIER_GEN
- Need a DNS domain name? Generators#DNS_DOMAIN_NAME
- Need UTF8? Generators#UTF_8_GEN
- Time? Generators#TIMESTAMP_GEN

Anatomy of a Property Test

Generate Data

- Hundreds of Cassandra generators exist today!
- Need Types? AbstractTypeGenerators#typeGen()
- Need tables? TableMetadataBuilder#build()
- Need data for a type? AbstractTypeGenerators#getTypeSupport

Bringing it all together

```
// Create property builder using the "qt" function
// Allows overriding seed/examples/etc.
qt()
```

Bringing it all together

```
// Select the Gen as input to the test
qt().forAll(Gens.ints().all())
```

Bringing it all together

```
// Add the test to the builder's "check" method
qt().forAll(Gens.ints().all()).check(value -> {
  assertThat(value / value).isEqualTo(1);
});
```

Bringing it all together

```
// Add the test to the builder's "check" method
qt().forAll(Gens.ints().all()).check(value -> {
  assertThat(value / value).isEqualTo(1);
});
```

```
accord.utils.Property$PropertyError: Property error detected:
Seed = 3448125481938895569
Examples = 1000
Pure = true
Error: / by zero
Values:
    0 = 0: java.lang.Integer
    at accord.utils.Property$SingleBuilder.checkInternal(Property.java:251)
    at accord.utils.Property$SingleBuilder.check(Property.java:235)
    at org.apache.cassandra.cql3.validation.operations.DemoTest.test7(DemoTest.java:151)
Caused by: java.lang.ArithmeticException: / by zero
    at org.apache.cassandra.cql3.validation.operations.DemoTest.lambda$test7$2(DemoTest.java:152)
    at accord.utils.Property$SingleBuilder.checkInternal(Property.java:247)
    ... 32 more
```

Bringing it all together

```
// Rerun the failing case by controlling the seed
qt().withSeed(3448125481938895569L).forAll(Gens.ints().all()).check(value -> {
  assertThat(value / value).isEqualTo(1);
});
```

Example

Serialize / Deserialize

```
qt().forAll(token()).check(token -> {
  var p = token.getPartitioner();
  var comparable = asComparableBytes(token);
  Token read = fromComparableBytes(p, comparable);
  assertThat(read).isEqualTo(token);
});
```

See org.apache.cassandra.dht.IPartitionerTest#byteCompareSerde

Example

Serialize / Deserialize Continued

```
DataOutputBuffer output = new DataOutputBuffer();
qt().forAll(token()).check(token -> {
  for (MessagingService.Version version : supportedVersions())
    // see IVersionedSerializers.testSerde
    testSerde(output, Token.compactSerializer, token, version.value);
});
```

See org.apache.cassandra.dht.TokenTest#serde

Example

Write / Read

```
createTable("CREATE TABLE %s(pk int, v int, PRIMARY KEY(pk))");
execute("INSERT INTO %s(pk, v) VALUES (?, ?)", 0, 0);
assertRows(execute("SELECT * FROM %s WHERE pk=?", 0),
           row(0, 0));
```

Example

Write / Read

```
qt().check(rs -> {
  TableMetadata metadata = createTable(rs, createKeyspace(rs).name);
  Mutation mutation = createMutation(rs, metadata);
  execute(mutation);

  Select select = select(mutation);
  assertRows(execute(select),
             rows(mutation));
});
```

Stateful Property

Stateful Property

- Just a property that has… state
- Commonly represented with a “model”
- Don’t be scared… they don’t bite!

Example

Create / Read / Update / Delete

```
enum Action { Create, Read, Update, Delete }
Gen<Action> ACTION_GEN = Gens.enums().all(Action.class);
qt().check(rs -> {
  TreeMap<Integer, Integer> model = new TreeMap<>();
  RangeTree<Integer, Integer, Integer> tree = intTree();
  for (int i = 0; i < 1000; i++) {
    Gen<Action> actionGen = model.isEmpty() ? constant(Action.Create) : ACTION_GEN;
    switch (actionGen.next(rs)) {
      case Create: doCreate(rs, model, tree); break;
      case Read:   doRead(rs, model, tree); break;
      case Update: doUpdate(rs, model, tree); break;
      case Delete: doDelete(rs, model, tree); break;
    }
  }
});
```

See org.apache.cassandra.utils.RTreeFuzzTest#mapLike

Example

Create / Read / Update / Delete

```
void doCreate(RandomSource rs, TreeMap<Integer, Integer> model, RangeTree<Integer, Integer, Integer> tree) {
  int key = rs.nextInt();
  while (model.containsKey(key))
    key = rs.nextInt();
  int value = rs.nextInt();
  model.put(key, value);
  tree.add(key, value);
  assertThat(tree.size()).isEqualTo(model.size());
}

void doRead(RandomSource rs, TreeMap<Integer, Integer> model, RangeTree<Integer, Integer, Integer> tree) {
  int key = rs.pick(model.keySet());
  int expected = model.get(key);
  assertThat(tree.get(key)).isEqualTo(Arrays.asList(expected));
}
```

See org.apache.cassandra.utils.RTreeFuzzTest#mapLike

Example

SAI Index

```
protected void test(AbstractType<?> type, Gen<Gen<ByteBuffer>> distribution) {
  createTable("CREATE TABLE %s (pk int PRIMARY KEY, value " + type.asCQL3Type() + ')');
  createIndex("CREATE INDEX ON %s(value) USING 'sai'");
  qt().check(rs -> {
      truncateTable();
      Gen<ByteBuffer> gen = distribution.next(rs);
      Map<ByteBuffer, IntArrayList> termIndex = new TreeMap<>();

      for (int i = 0; i < 1000; i++) {
        ByteBuffer term = gen.next(rs);
        execute("INSERT INTO %s (pk, value) VALUES (?, ?)", i, term);
        termIndex.computeIfAbsent(term, ignore -> new IntArrayList()).addInt(i);
      }

      for (var e : termIndex.entrySet()) {
        ByteBuffer term = e.getKey();
        var expected = e.getValue();
        assertEquals(execute("SELECT pk, value FROM %s WHERE value=?", term), expected);
      }
  });
}
```

Bias

Why Add Bias?

- Most types have a large domain and/or similar values
  - If you work with 1, you likely work with 2… but do you work with -1? 0?
- Relationships between data might not be visible without explicit bias
  - ipv4 can convert to ipv6; both addresses are equal… just not in C*
- Edge cases are unlikely

Back to RangeTree

Devils in the details…

- When small, basically a List
- Once it grows larger, it “splits”

Back to RangeTree

Devils in the details...

```
enum Action { Create, Read, Update, Delete }
Gen<Action> ACTION_GEN = Gens.enums().all(Action.class);
qt().check(rs -> {
  TreeMap<Integer, Integer> model = new TreeMap<>();
  RangeTree<Integer, Integer, Integer> tree = intTree();
  for (int i = 0; i < 1000; i++) {
    Gen<Action> actionGen = model.isEmpty() ? constant(Action.Create) : ACTION_GEN;
    switch (actionGen.next(rs)) {
      case Create: doCreate(rs, model, tree); break;
      case Read:   doRead(rs, model, tree); break;
      case Update: doUpdate(rs, model, tree); break;
      case Delete: doDelete(rs, model, tree); break;
    }
  }
});
```

See org.apache.cassandra.utils.RTreeFuzzTest#mapLike

Back to RangeTree

Devils in the details...

```
enum Action { Create, Read, Update, Delete }
Gen<Gen<Action>> ACTION_GEN = Gens.enums().allMixedDistribution(Action.class);
qt().check(rs -> {
  Gen<Action> gen = ACTION_GEN.next(rs);
  TreeMap<Integer, Integer> model = new TreeMap<>();
  RangeTree<Integer, Integer, Integer> tree = intTree();
  for (int i = 0; i < 1000; i++) {
    Gen<Action> actionGen = model.isEmpty() ? constant(Action.Create) : gen;
    switch (actionGen.next(rs)) {
      case Create: doCreate(rs, model, tree); break;
      case Read:   doRead(rs, model, tree); break;
      case Update: doUpdate(rs, model, tree); break;
      case Delete: doDelete(rs, model, tree); break;
    }
  }
});
```

See org.apache.cassandra.utils.RTreeFuzzTest#mapLike

INET Type

Equals isn’t always Equals…

- Supports IPV4 and IPV6
- IPV4 can convert to IPV6
  - IPV6 address is the same as the IPV4 address; they are functionally equal!

INET Type

Equals isn't always Equals...

```
@Test
public void ipv4Overlap() {
  test(InetAddressType.instance, rs -> {
    var uniqueIpv4 = Gens.lists(ipv4)
                         .unique()
                         .ofSizeBetween(5,100)
                         .next(rs);
    var uniqueIpv4AsIpv6 = uniqueIpv4.stream()
                                     .map(this::mapIpv4ToIpv6)
                                     .collect(Collectors.toList());
    return r -> r.pick(r.nextBoolean() ? uniqueIpv4 : uniqueIpv4AsIpv6);
  });
}
```

See https://issues.apache.org/jira/browse/CASSANDRA-19492

INET Type

Equals isn’t always Equals...

```
protected void test(AbstractType<?> type, Gen<Gen<ByteBuffer>> distribution) {
  createTable("CREATE TABLE %s (pk int PRIMARY KEY, value " + type.asCQL3Type() + ')’);
  createIndex("CREATE INDEX ON %s(value) USING 'sai'");
  qt().check(rs -> {
      truncateTable();
      Gen<ByteBuffer> gen = distribution.next(rs);
      Map<ByteBuffer, IntArrayList> termIndex = new TreeMap<>();

      for (int i = 0; i < 1000; i++) {
        ByteBuffer term = gen.next(rs);
        execute("INSERT INTO %s (pk, value) VALUES (?, ?)", i, term);
        termIndex.computeIfAbsent(term, ignore -> new IntArrayList()).addInt(i);
      }

      for (var e : termIndex.entrySet()) {
        ByteBuffer term = e.getKey();
        var expected = e.getValue();
        assertEquals(execute("SELECT pk, value FROM %s WHERE value=?", term), expected);
      }
  });
}
```

Bringing it all together

Stateful edition

```
stateful()
```

Bringing it all together

Stateful edition

```
stateful()
.check(commands(() -> State::new, Sut::new)
   ...
   .build());
```

Bringing it all together

Stateful edition

```
stateful()
.check(commands(() -> State::new, Sut::new)
  .add(this::create)
  .add(this::read)
  .add(this::readByKey)
  .add(this::readByRange)
  .add(Iterate.instance)
  .add(Clear.instance)
  .addAllIf(state -> !state.isEmpty(),
            b -> b.add(this::readExisting)
                  .add(this::readKeyFromRangeExisting)
                  .add(this::readRangeExisting)
                  .add(this::updateExisting)
                  .add(this::delete))
  .build());
```

Bringing it all together

Stateful edition

```
record RangeRead(Range range) implements Command<State, Sut, Result> {
  @Override
  public Result apply(State state) {
    return state.scan(range);
  }
  @Override
  public Result run(Sut sut) {
    return sut.tree.search(range);
  }
  @Override
  public String detailed(State state) {
    return "Range Read(" + range + ")";
  }
  @Override
  public void checkPostconditions(State state, Result expected,
                                  Sut sut,     Result actual) {
    assertThat(actual).isEqualTo(expected);
  }
}
```

Bringing it all together

Stateful edition

```
accord.utils.Property$PropertyError: Property error detected:
Seed = 3448148273639475646
Examples = 500
Pure = true
Error:

    expected: []
     but was: [(37492,37803]=5]
Steps: 1000
Values:
    State: State{sizeTarget=389, numChildren=5}: StatefulRangeTreeTest.State
    History:
        1: Create((14490,14554], 0)
        2: Token Read(46197)
        3: Clear(size=1)
        4: Create((37492,37803], 5)
        5: Range Read((37492,37803])

Expected :[]
Actual   :[(37492,37803]=5]
<Click to see difference>
    at StatefulRangeTreeTest.test(StatefulRangeTreeTest.java:72)

Caused by: org.opentest4j.AssertionFailedError:
expected: []
 but was: [(37492,37803]=5]
    at StatefulRangeTreeTest$RangeRead.checkPostconditions(StatefulRangeTreeTest.java:188)
    at StatefulRangeTreeTest$RangeRead.checkPostconditions(StatefulRangeTreeTest.java:173)
```

Simulation Property

Simulation Property

- The test controlling running of one or more systems
  - Thread scheduling, message delivery, time, etc.
- Can simulate as much as you need, or as little
  - Need all of Cassandra? The Simulator is your friend
  - Need just messaging? Thread scheduling? There are cheaper options

Simulation Property

Repair Happy Path

```
qt().check(rs -> {
  // creates a multi-node C* mock cluster with simulated threads/messaging
  Cluster cluster = new Cluster(rs);
  for (int example = 0; example < 100; example++) {
    Cluster.Node coordinator = cluster.nextCoordinator(rs);
    RepairCoordinator repair = coordinator.repair(repairOption(rs, coordinator));
    // start the repair, this enqueues threads/messages
    repair.run();
    cluster.processAll(); // simulate all tasks until done
    assertSuccess(repair);
  }
});
```

See org.apache.cassandra.repair.HappyPathFuzzTest#happyPath

Simulation Property

How is this done?

- Simulator changed many core systems to be interfaces...
- ExecutorFactory - interface for creating threads
  - SimulatedExecutorFactory enqueues all work into a PriorityQueue
- Clock - interface for accessing wall-time and progress ticks
  - SimulatedExecutorFactory defines time as the progress of work
- MessageDelivery - interface for sending internode messages
  - SimulatedMessageDelivery allows running with and without faults
- File - path within a FileSystem (interface) for disk access
  - ListenableFileSystem adds pre/post callbacks to all FileSystem methods

Simulation Property

Repair Happy Path

```
Cluster(RandomSource rs) {
  this.globalExecutor = new SimulatedExecutorFactory(rs.fork());
  IPartitioner partitioner = PARTITIONER_GEN.next(rs);
  this.tokenGen = CassandraGenerators.token(partitioner)
                                     .filter(t -> !tokens.contains(t));
  int numTokens = NUM_TOKEN_GEN.nextInt(rs);
  int numNodes = NUM_NODE_GEN.nextInt(rs);
  for (int i = 0; i < numNodes; i++) {
    InetAddressAndPort address = addressGen.next(rs);
    List<Token> tokens = Gens.lists(tokenGen)
                             .unique()
                             .ofSize(numTokens).next(rs);
    Node node = new Node(address, tokens, new Messaging(address))
    this.nodes.put(node.address, node);
  }
}
```

See org.apache.cassandra.repair.HappyPathFuzzTest#happyPath

Simulation Property

Repair Happy Path

```
private class Messaging extends SimulatedMessageDelivery {
  private Messaging(InetAddressAndPort address) {
    super(address,
          Cluster.this::action, // What we can do to the msg: DELIVER/DROP
          SimulatedMessageDelivery.randomDelay(rs.fork()),
          (to, message) -> unorderedScheduled.execute(() -> nodes.get(to).receiver.recieve(message)),
          unorderedScheduled::schedule,
          failures::add);
  }
}
```

See org.apache.cassandra.repair.HappyPathFuzzTest#happyPath

Faulty Simulation!

Why be happy when we can live with chaos?

- Scheduling
  - Can vary the delay between tasks, including short/long delays
  - Change ordering
- Messaging
  - Partition, Drop, and Delay messages
- Disk
  - Read-only, corrupt, delay

Simulation Property

Repair Slow Messages

```
qt().check(rs -> {
  // creates a multi-node C* mock cluster with simulated threads/messaging
  Cluster cluster = new Cluster(rs);
  enableMessageFaults(cluster);
  for (int example = 0; example < 100; example++) {
    Cluster.Node coordinator = cluster.nextCoordinator(rs);
    RepairCoordinator repair = coordinator.repair(repairOption(rs, coordinator));
    // start the repair, this enqueues threads/messages
    repair.run();
    cluster.processAll(); // simulate all tasks until done
    assertSuccess(repair);
  }
});
```

See org.apache.cassandra.repair.SlowMessageFuzzTest#slowMessages

Simulation Property

Failed Repair Stages

```
qt().check(rs -> {
  // creates a multi-node C* mock cluster with simulated threads/messaging
  Cluster cluster = new Cluster(rs);
  enableMessageFaults(cluster);
  for (int example = 0; example < 100; example++) {
    Cluster.Node coordinator = cluster.nextCoordinator(rs);
    RepairCoordinator repair = coordinator.repair(repairOption(rs, coordinator));
    // start the repair, this enqueues threads/messages
    repair.run();
    InetAddressAndPort failingAddress = rs.pick(repair.participants);
    // adds repair specific faults in the different stages
    enableRepairFaults(failingAddress);
    cluster.processAll(); // simulate all tasks until done
    assertFailedassertSuccess(repair);
  }
});
```

See org.apache.cassandra.repair.FailedAckTest#failedAck

Closing remarks

- Not starting from scratch
  - a ton of resources already exist to help you
- You can replace your tests with property tests; it is not that scary
- Tests can evolve and grow with the system; not static snapshots
- “Model” isn’t scary… 80/20!

Q & A