SparkFuzz:
Searching Correctness Regressions in Modern Query Engines

Bogdan Ghit, **Nicolas Poggi**, Josh Rosen, Reynold Xin, and Peter Boncz*

June 19 - DBTest 2020
UNIFIED DATA ANALYTICS PLATFORM

DATA SCIENCE WORKSPACE

UNIFIED DATA SERVICE

ENTERPRISE CLOUD SERVICE
Introduction

Apache Spark

Fast and expressive data processing engine
- distributed computing
- rich APIs
  - including SQL
- large community

Started at UC Berkeley in 2009
- 2010 - open sourced
- 2014 - top level project
- 2020 - v3 released (10 years!)

June 2002 v 3.0.0 released

3500+ resolved tickets

- Spark SQL: 46%
- MLlib/ML: 16%
- Spark Core: 16%
- Structured Streaming: 12%
- Tests/Docs: 6%
- PySpark: 7%
- Others: 9%

SparkFuzz proposal

1. Leverage fuzz testing techniques
   a. to complement SQL testing
   b. automate bug discovery
2. Design of a toolkit for SQL engines
   a. model for randomized
      i. DDL, data, and queries
   b. A runner and evaluator
3. Applicability of coverage metrics
   a. as test stop gaps
   b. reducing time (and costs)
   c. enabling more testing dimensions
DDL and data generation

Automated dataset generation

- by randomly sampling
  - supported data types
  - parameter ranges

- Producing valid schemas

- Populating datasets
Recursive query model w/ a probabilistic profile

Operators and features annotated with:

Independent weights
- Optional clauses

Inter-dependent weights
- Join types
- Select functions

SQL Query

WITH
UNION
SELECT
WHERE
GROUP BY
ORDER BY
JOIN
FROM

Query
Clause
Expression
Functions
Constant
Alias
Column
Table
Query and regression example

Query produced in a small dataset with 2 tables of 5x5 size

```
SELECT COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3) AS int_col,
       IF(NULL, VARIANCE(COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3)),
          COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3)) AS int_col_1,
       STDDEV(t2.double_col_2) AS float_col,
       COALESCE(MIN((t1.smallint_col_3) - (COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3))),
                 COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3)) AS int_col_2
FROM table_4 t1
INNER JOIN table_4 t2 ON (t2.timestamp_col_7) = (t1.timestamp_col_7)
WHERE (t1.smallint_col_3) IN (CAST('0.04' AS DECIMAL(10,10)), t1.smallint_col_3)
GROUP BY COALESCE(t2.smallint_col_3, t1.smallint_col_3, t2.smallint_col_3)
```

- Within 10 queries, this query triggered an exception
- Related to COALESCE flattening
Correctness regression example [SPARK-16633]

Using constant input values breaks the the LEAD function

```sql
SELECT (t1.decimal0803_col_3) / (t1.decimal0803_col_3) AS decimal_col,
       CAST(696 AS STRING) AS char_col, t1.decimal0803_col_3,
       (COALESCE(CAST('0.02' AS DECIMAL(10,10)),
                  CAST('0.47' AS DECIMAL(10,10)),
                  CAST('-0.53' AS DECIMAL(10,10)))) +
       (LEAD(-65, 4) OVER (ORDER BY (t1.decimal0803_col_3) / (t1.decimal0803_col_3),
                         CAST(696 AS STRING))) AS decimal_col_1,
       CAST(-349 AS STRING) AS char_col_1
FROM table_16 t1
WHERE (943) > (889)
```

- Spark [1.0, 696, -871.81, -64.98, -349]
- PostgreSQL [1.0, 696, -871.81, NULL, -349]
Query operator coverage analysis

In 15m (500 queries), reaches near max coverage
Continuous Integration pipeline

- Impact
- Scope
- Correlation
- Confirm?

- Minimize
- Drill-down
- Profile
- Compare
- Validate

Events

Failure

Regression

Correctness

Performance

Root-cause

Alert

Re-test

SparkFuzz
Conclusion and future work

- Prevented SQL correctness errors reaching production
  - complementing the testing practices

- Runtime operator coverage metrics found applicable
  - For testing code changes rapidly
  - With a degree of coverage

- Future work
  - Improve the metric coverage to include operator chaining
  - Update the model generation to use Spark AST grammar directly
Thanks, questions?

Bogdan Ghit, Nicolas Poggi, Josh Rosen, Reynold Xin, and Peter Boncz

Feedback: Nicolas.Poggi@databricks.com