Correctness and Performance of Apache Spark SQL

Spark + AI Summit, London

October 4, 2018

databricks

About us

BOGDAN GHIT

Databricks, Software Engineer

- SQL performance optimizations

IBM T.J. Watson, Research Intern

- Bid advisor for cloud spot markets

Delft University of Technology, PhD in Computer Science

- Resource management in datacenters
- Performance of Spark, Hadoop

NICOLAS POGGI

Databricks, Performance Engineer

- Spark benchmarking

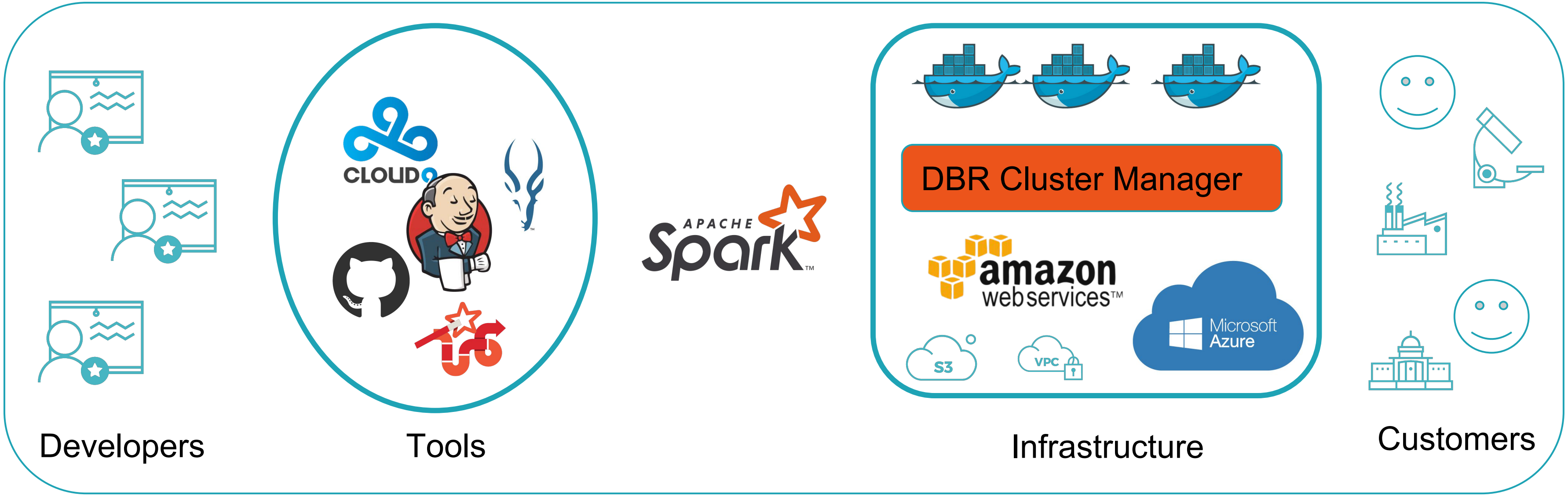
Barcelona Supercomputing - Microsoft Research Centre

- Lead researcher ALOJA project
- New architectures for Big Data

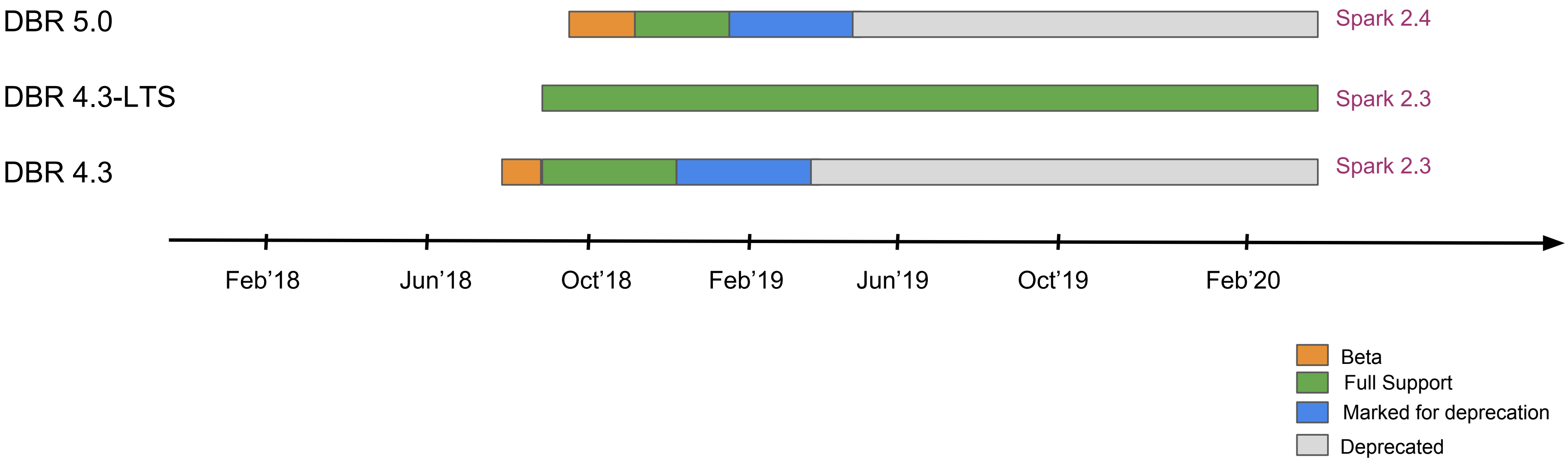
BarcelonaTech (UPC), PhD in Computer Architecture

- Autonomic resource manager for the cloud
- Web customer modeling

Databricks ecosystem

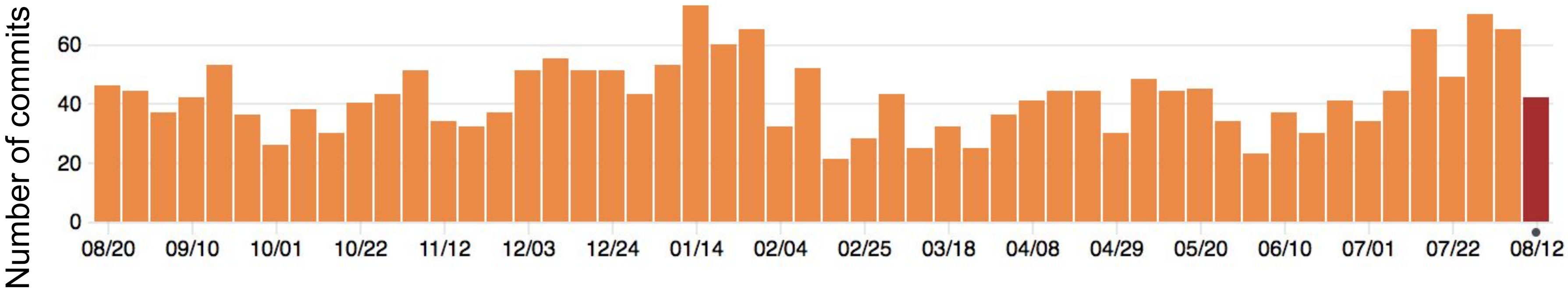


Databricks runtime (DBR) releases



Our goal is to make releases **automatic** and **frequent**

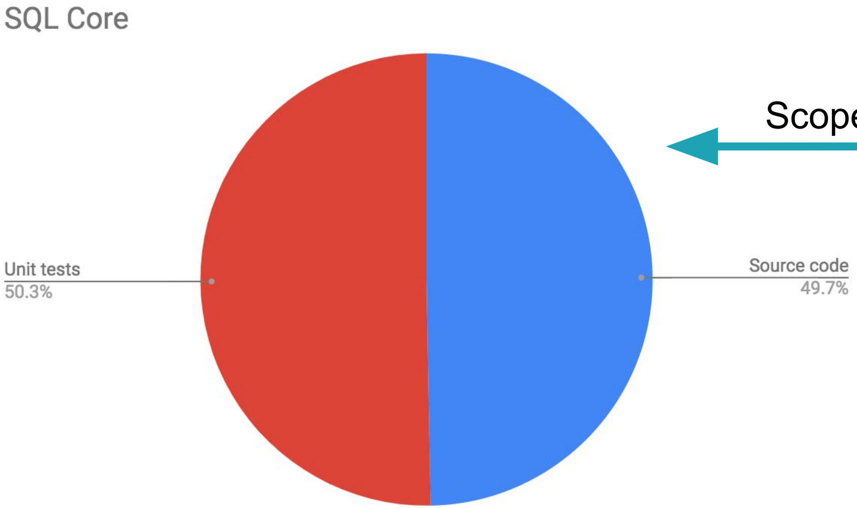
Apache Spark contributions



Hundreds of commits monthly to the Apache Spark project

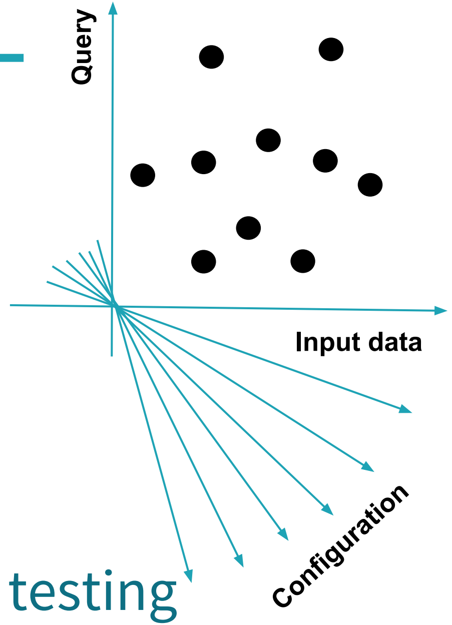
At this pace of development, **mistakes** are bound to happen

Where do these contributions go?



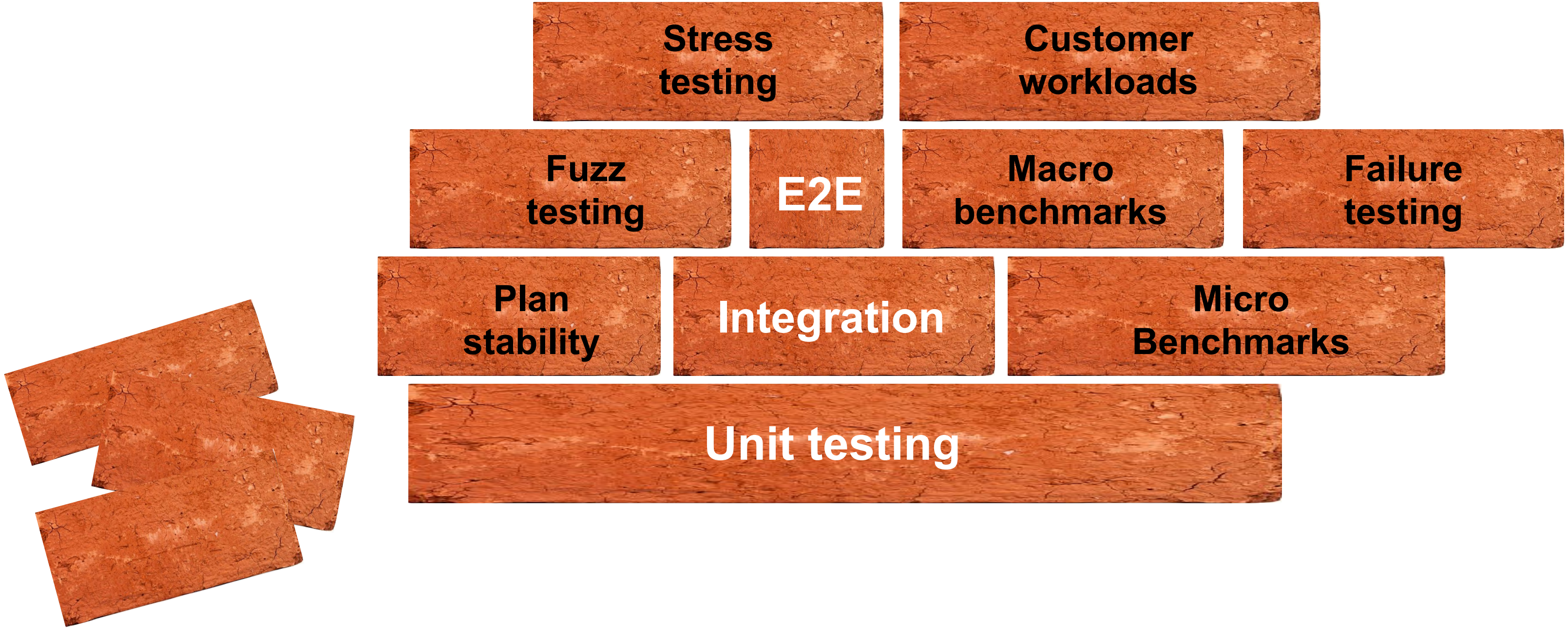
Scope of the testing

Over 200 built-in functions



Developers put a significant **engineering effort** in testing

Yet another brick in the wall



Stress testing

Customer workloads

Fuzz testing

E2E

Macro benchmarks

Failure testing

Plan stability

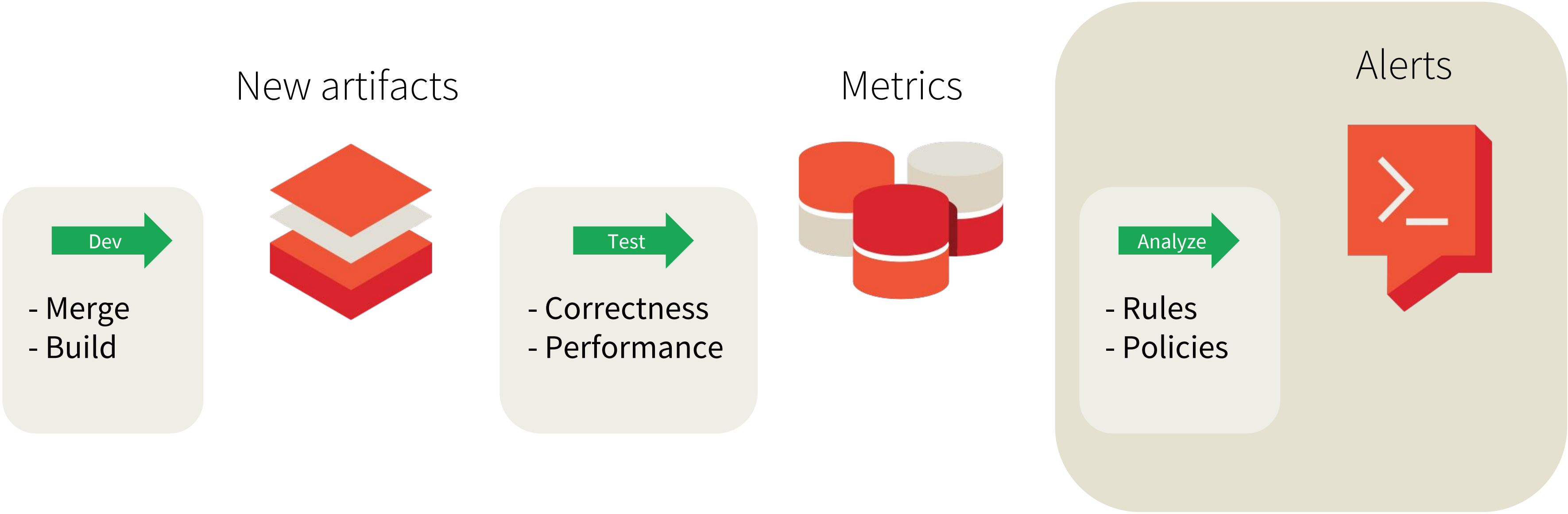
Integration

Micro Benchmarks

Unit testing

Unit testing ***is not enough*** to guarantee correctness and performance

Continuous Integration pipeline



Dev

- Merge
- Build

New artifacts

Test

- Correctness
- Performance

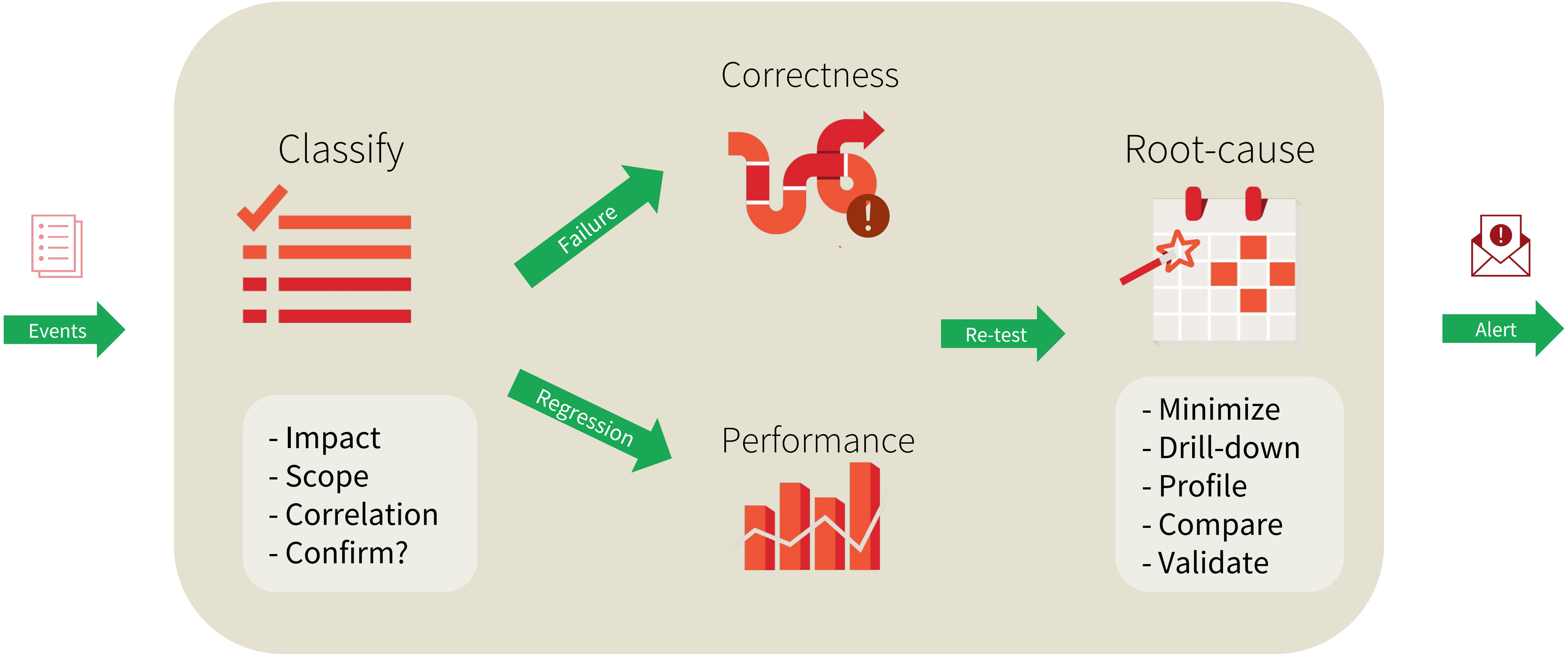
Metrics

Analyze

- Rules
- Policies

Alerts

Classification and alerting



Events

Classify

- Impact
- Scope
- Correlation
- Confirm?

Failure

Regression

Correctness

Performance

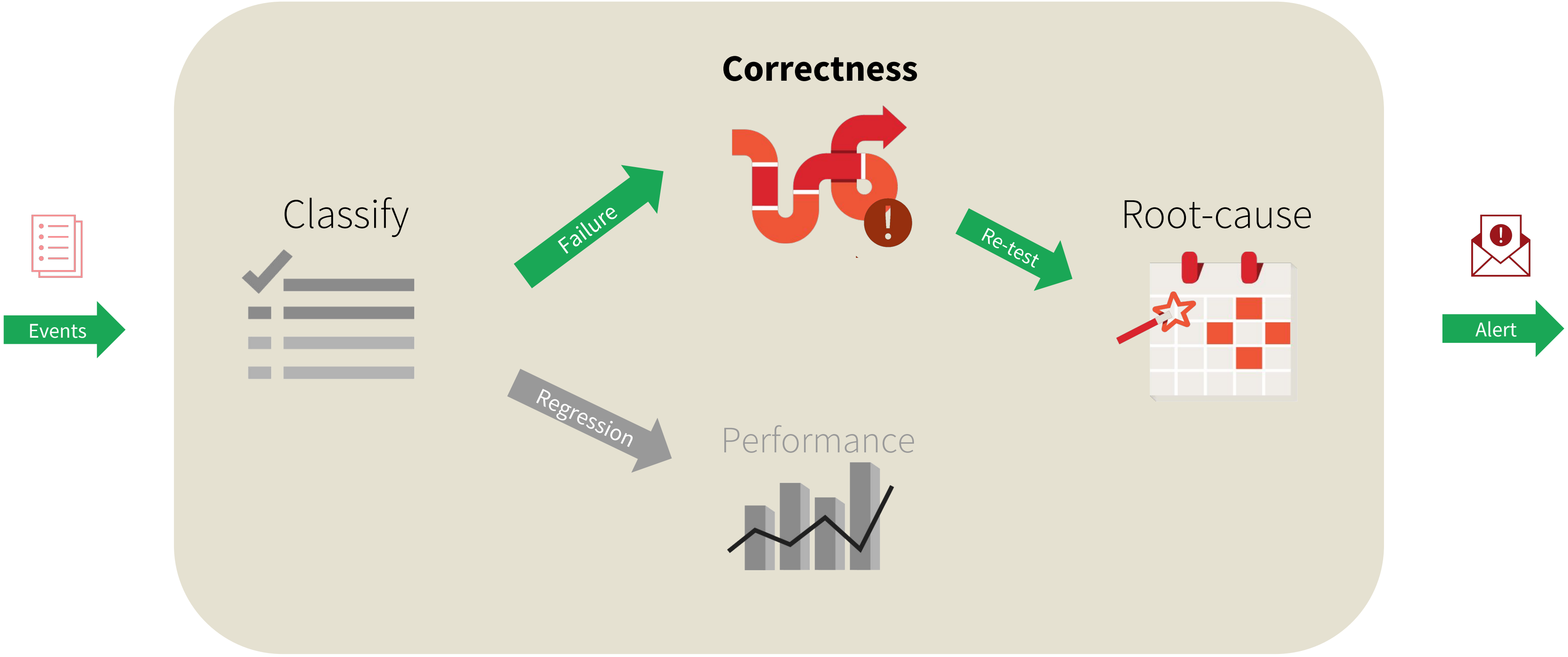
Re-test

Root-cause

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

Alert

Correctness



Events

Classify

Failure

Regression

Correctness

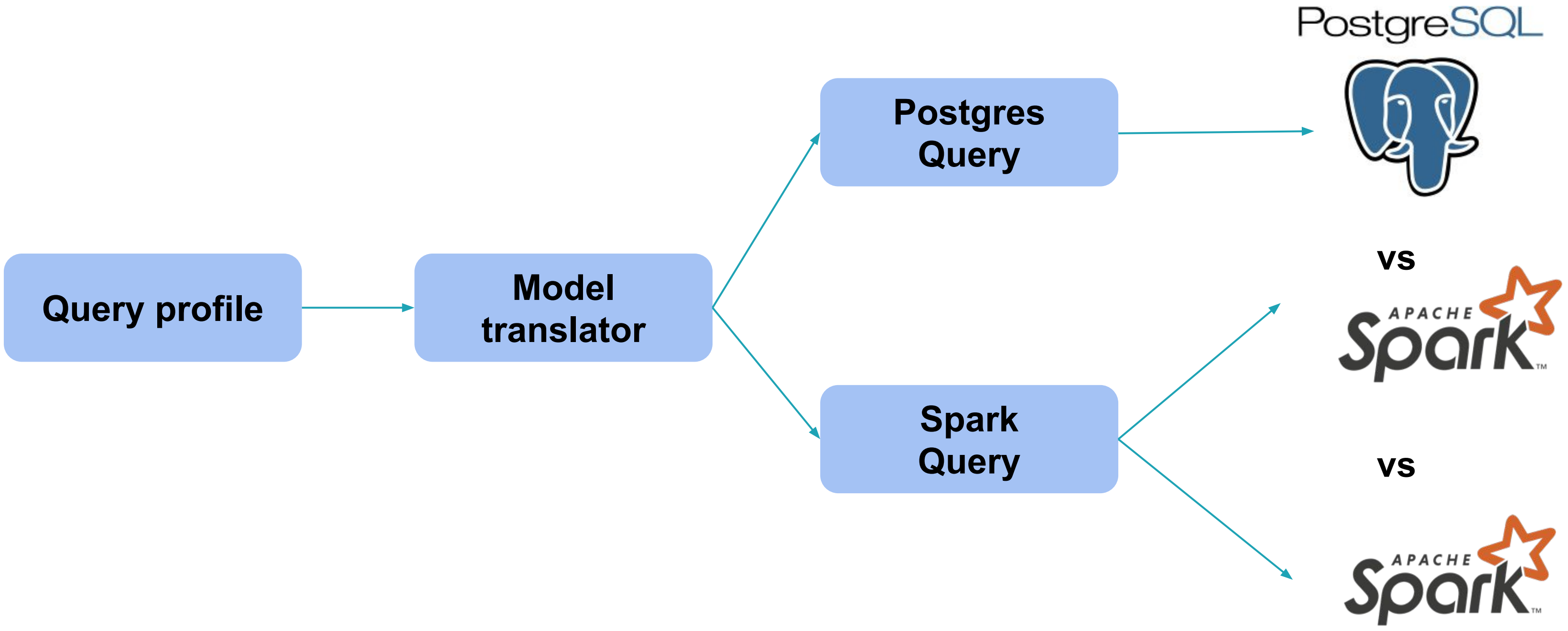
Performance

Re-test

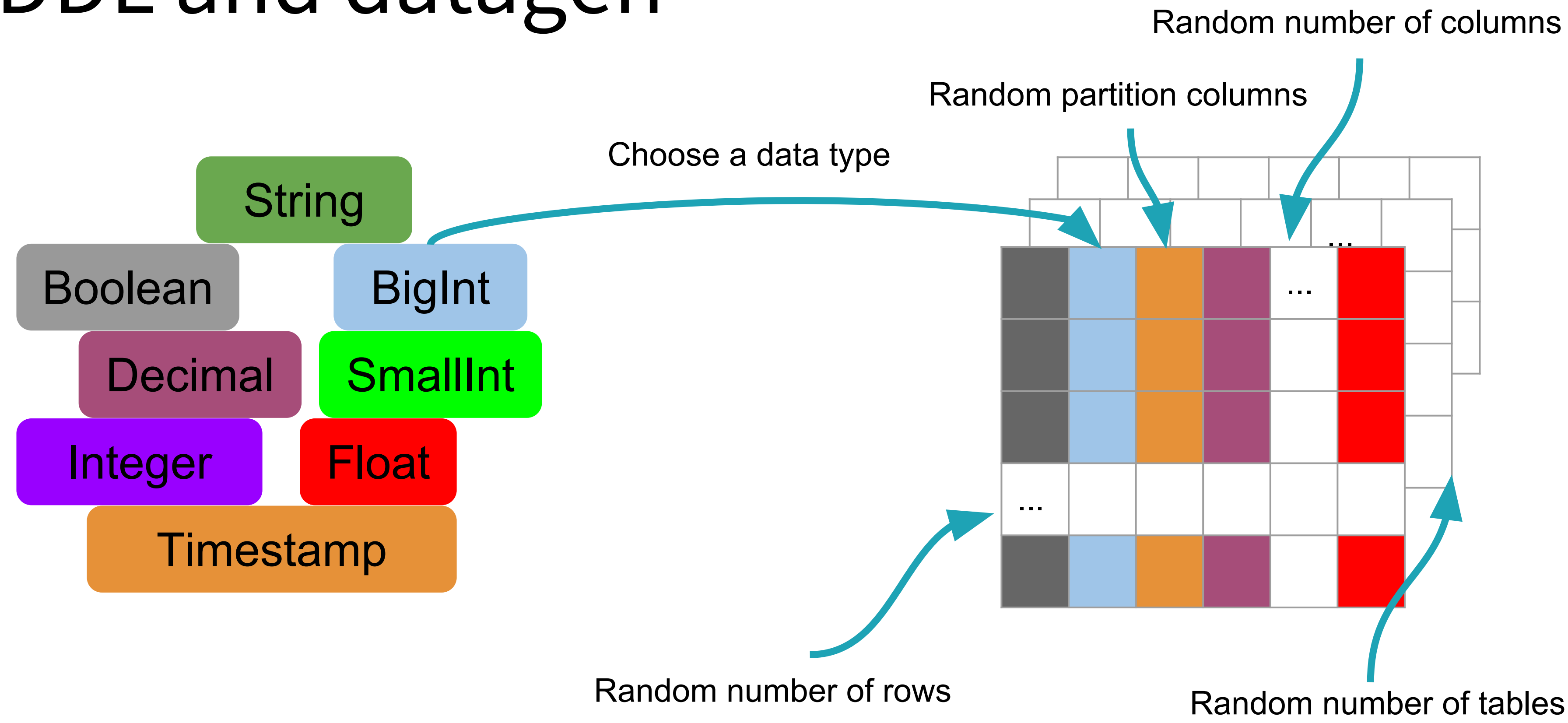
Root-cause

Alert

Random query generation



DDL and datagen



String

Boolean

BigInt

Decimal

SmallInt

Integer

Float

Timestamp

Choose a data type

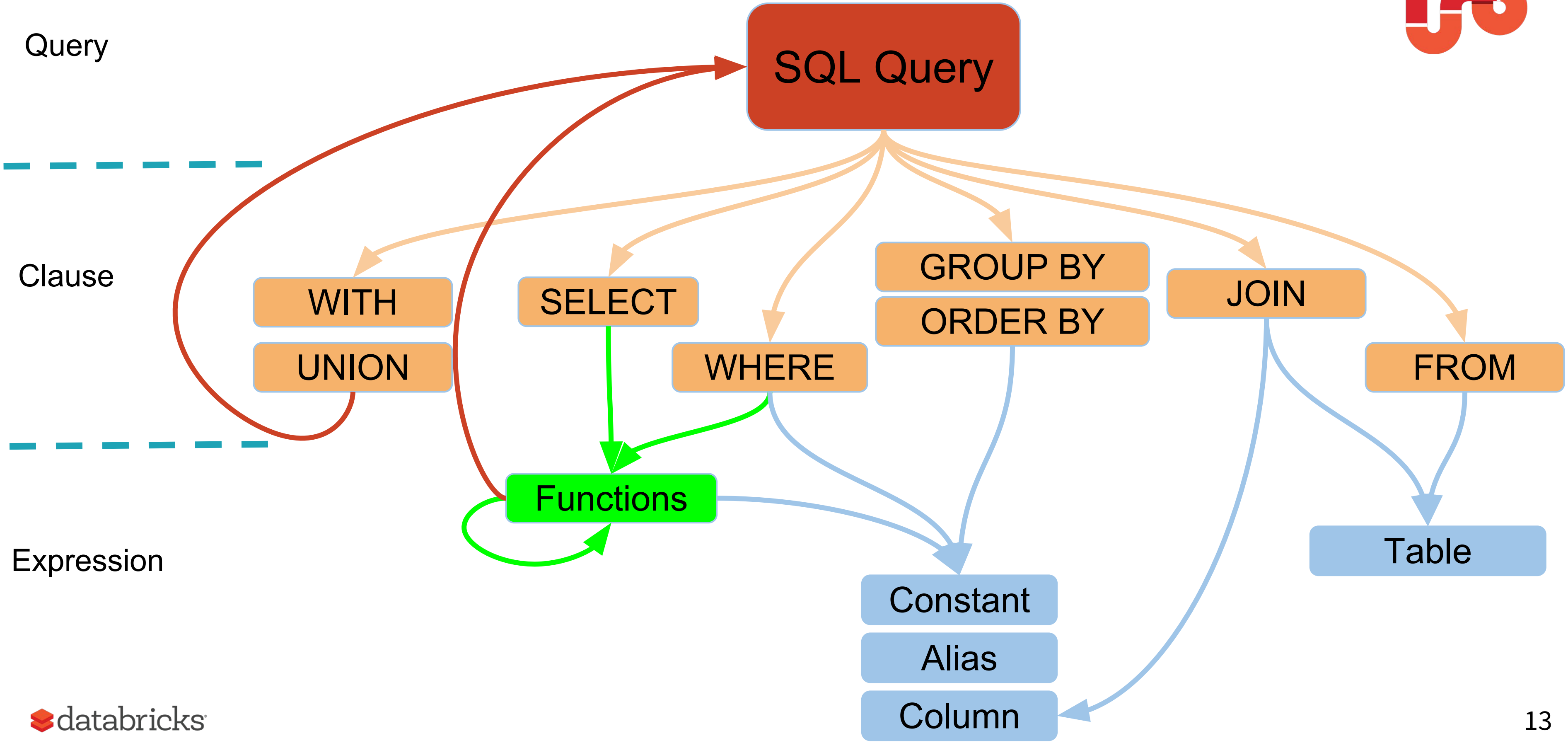
Random partition columns

Random number of columns

Random number of rows

Random number of tables

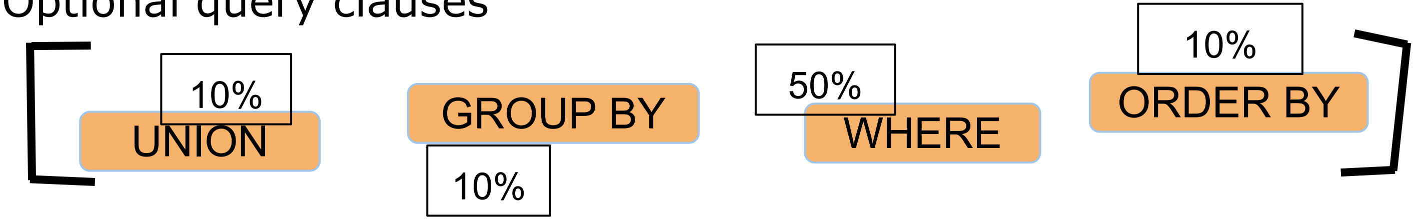
Recursive query model



Probabilistic query profile

Independent weights

- Optional query clauses



10%
UNION

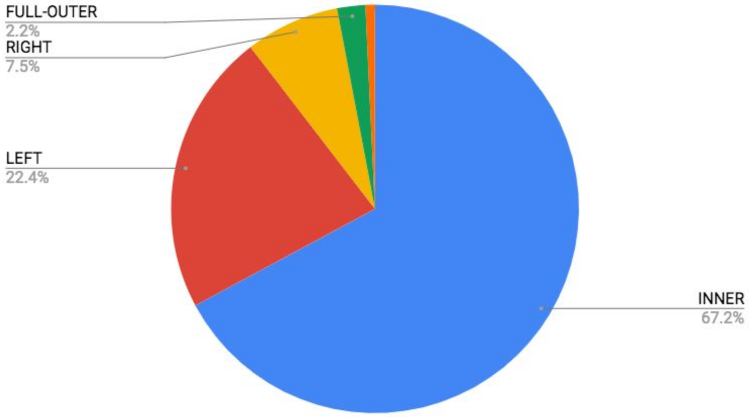
GROUP BY
10%

50%
WHERE

10%
ORDER BY

Inter-dependent weights

- Join types
- Select functions



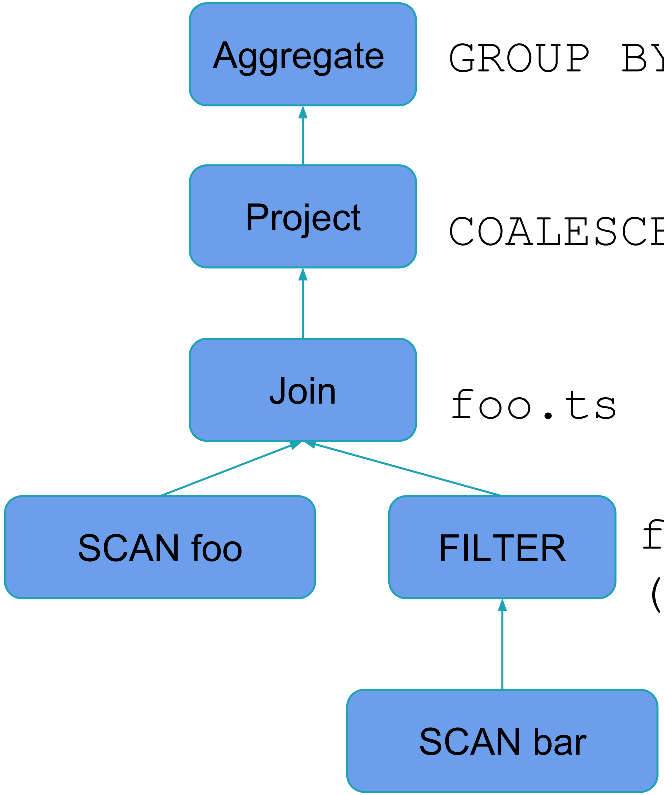
Coalesce flattening (1/4)

```
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),
       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)
```

Small dataset with 2 tables of 5x5 size
Within 10 randomly generated queries

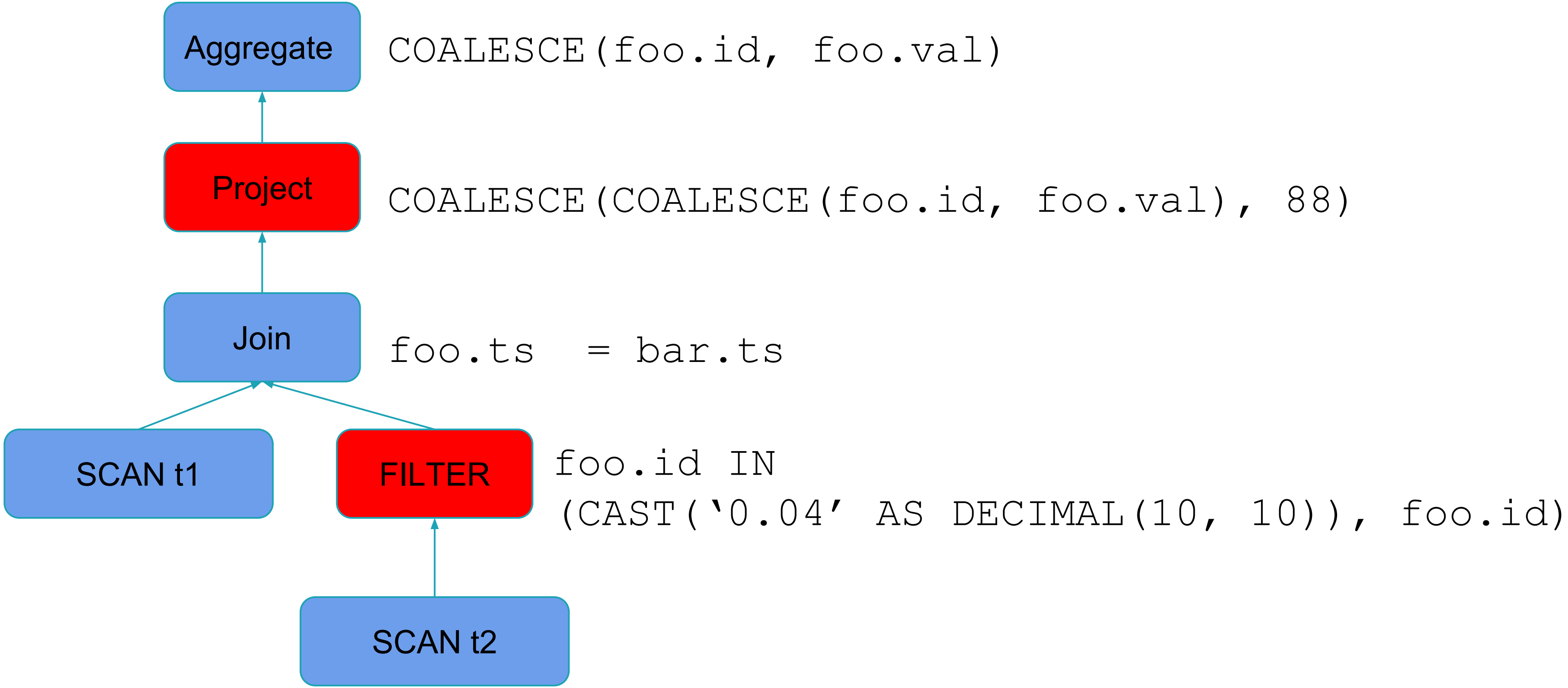
Error: Operation is in ERROR_STATE

Coalesce flattening (2/3)



```
GROUP BY COALESCE(foo.id, foo.val)

COALESCE(COALESCE(foo.id, foo.val), 88)

foo.ts  = bar.ts

foo.id IN
(CAST(‘0.04’ AS DECIMAL(10, 10)), foo.id)
```

Coalesce flattening (3/4)



Aggregate

```
COALESCE(foo.id, foo.val)
```

Project

```
COALESCE(COALESCE(foo.id, foo.val), 88)
```

Join

```
foo.ts  = bar.ts
```

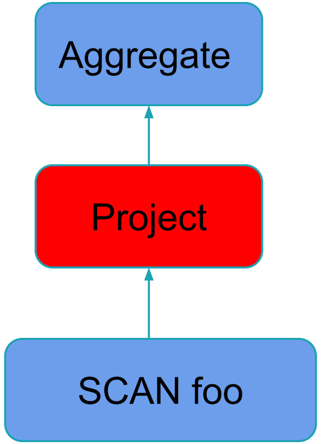
SCAN t1

FILTER

```
foo.id IN
(CAST(‘0.04’ AS DECIMAL(10, 10)), foo.id)
```

SCAN t2

Coalesce flattening (4/4)



Minimized query:

```
SELECT
   COALESCE(COALESCE(foo.id, foo.val), 88)
FROM foo
GROUP BY
   COALESCE(foo.id, foo.val)
```

Analyzing the error

- The optimizer flattens the nested coalesce calls
- The SELECT clause doesn’t contain the GROUP BY expression
- Possibly a problem with any GROUP BY expression that can be optimized

Lead function (1/3)

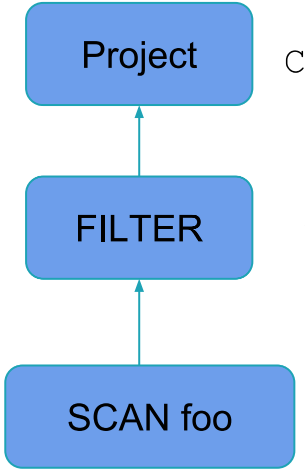
```
SELECT (t1.decimal0803_col_3) / (t1.decimal0803_col_3) AS decimal_col,
        CAST(696 AS STRING) AS char_col, t1.decimal0803_col_3,
       (COALESCE(CAST('0.02' AS DECIMAL(10,10)),
                 CAST('0.47' AS DECIMAL(10,10)),
                 CAST('-0.53' AS DECIMAL(10,10)))) +
       (LEAD(-65, 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)
```

Error: Column 4 in row 10 does not match:

[1.0, 696, -871.81, <<-64.98>>, -349] SPARK row

[1.0, 696, -871.81, <<None>>, -349] POSTGRESQL row

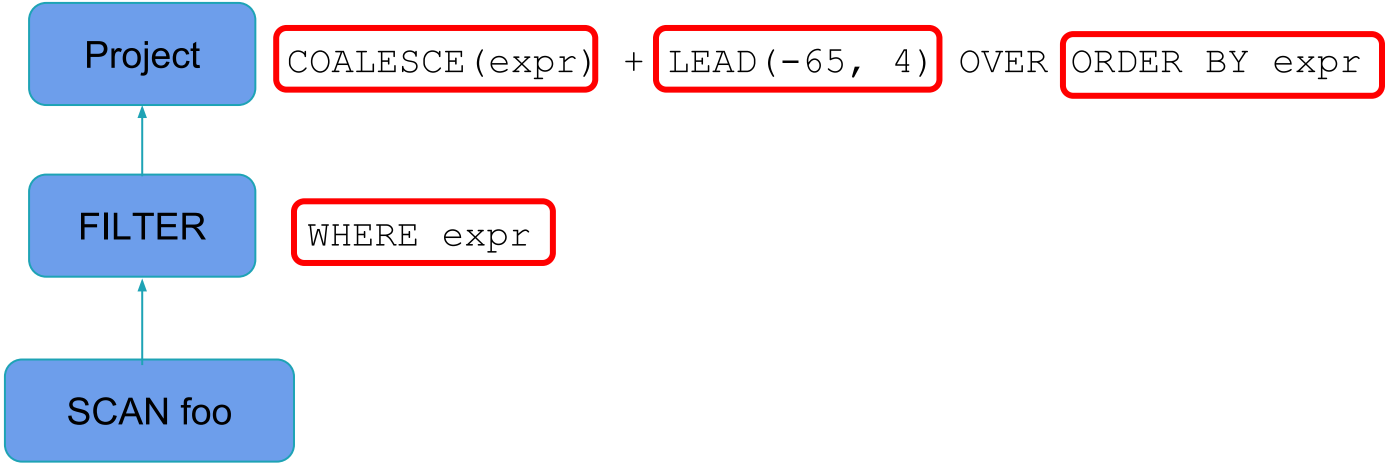
Lead function (2/3)



```
COALESCE(expr) + LEAD(-65, 4) OVER ORDER BY expr
```

```
WHERE expr
```

Lead function (3/3)



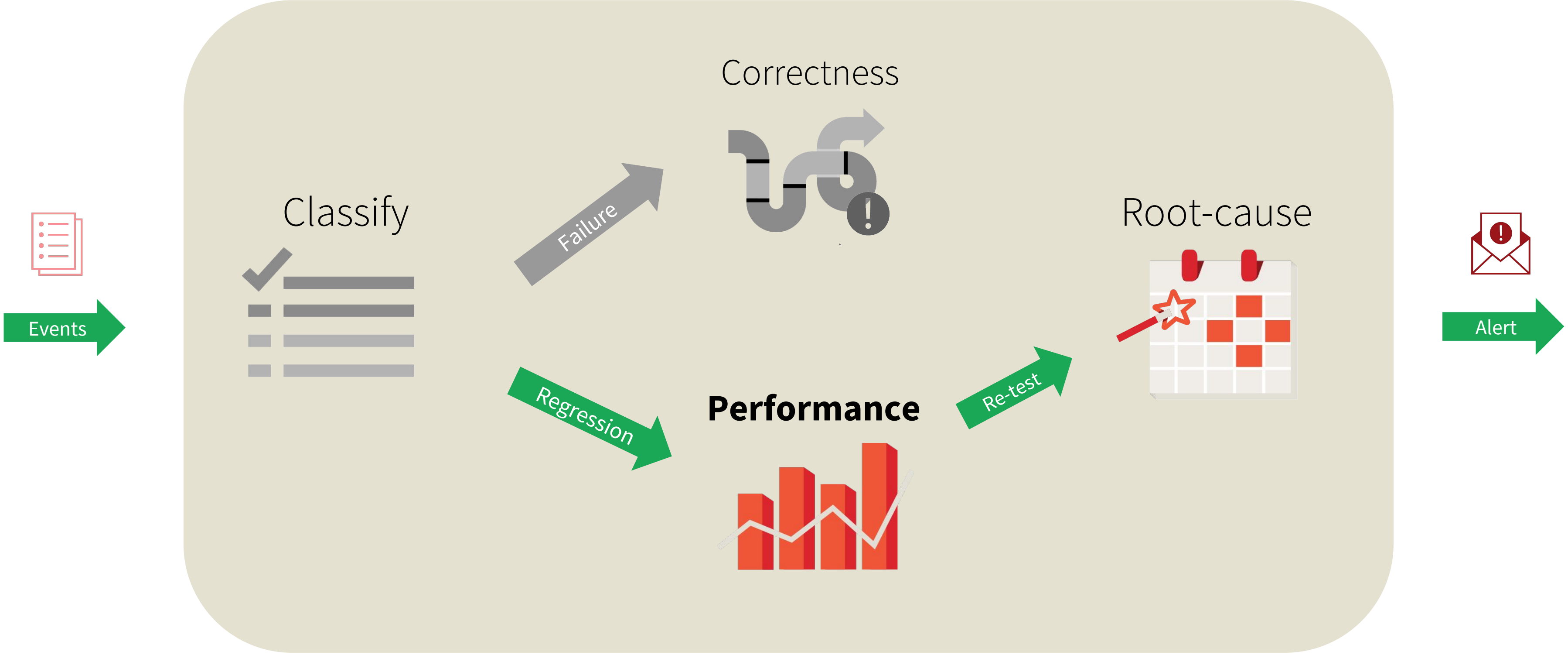
```
COALESCE(expr) + LEAD(-65, 4) OVER ORDER BY expr
```

```
WHERE expr
```

Analyzing the error

- Using constant input values breaks the behaviour of the LEAD function
- SC-16633: https://github.com/apache/spark/pull/14284

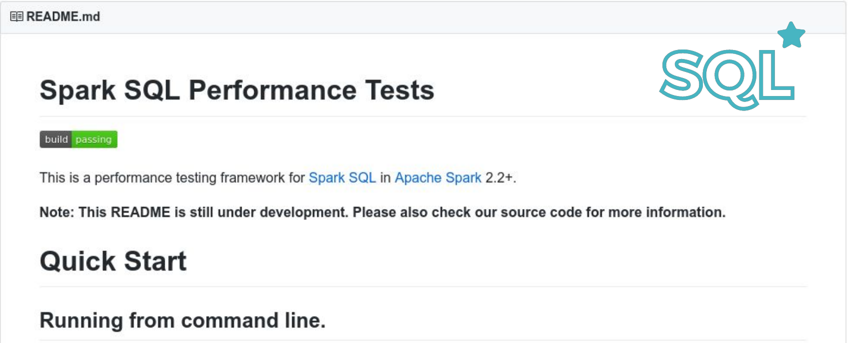
Performance



Benchmarking tools

- We use spark-sql-perf public library for TPC workloads
  - Provides datagen and import scripts
  - local, cluster, S3
  - Dashboards for analyzing results

- The Spark micro benchmarks
- And the async-profiler
  - to produce flamegraphs



README.md

Spark SQL Performance Tests

build passing

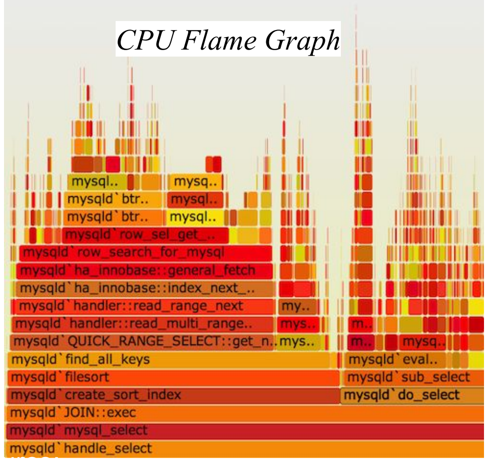
This is a performance testing framework for Spark SQL in Apache Spark 2.2+.

Note: This README is still under development. Please also check our source code for more information.

Quick Start

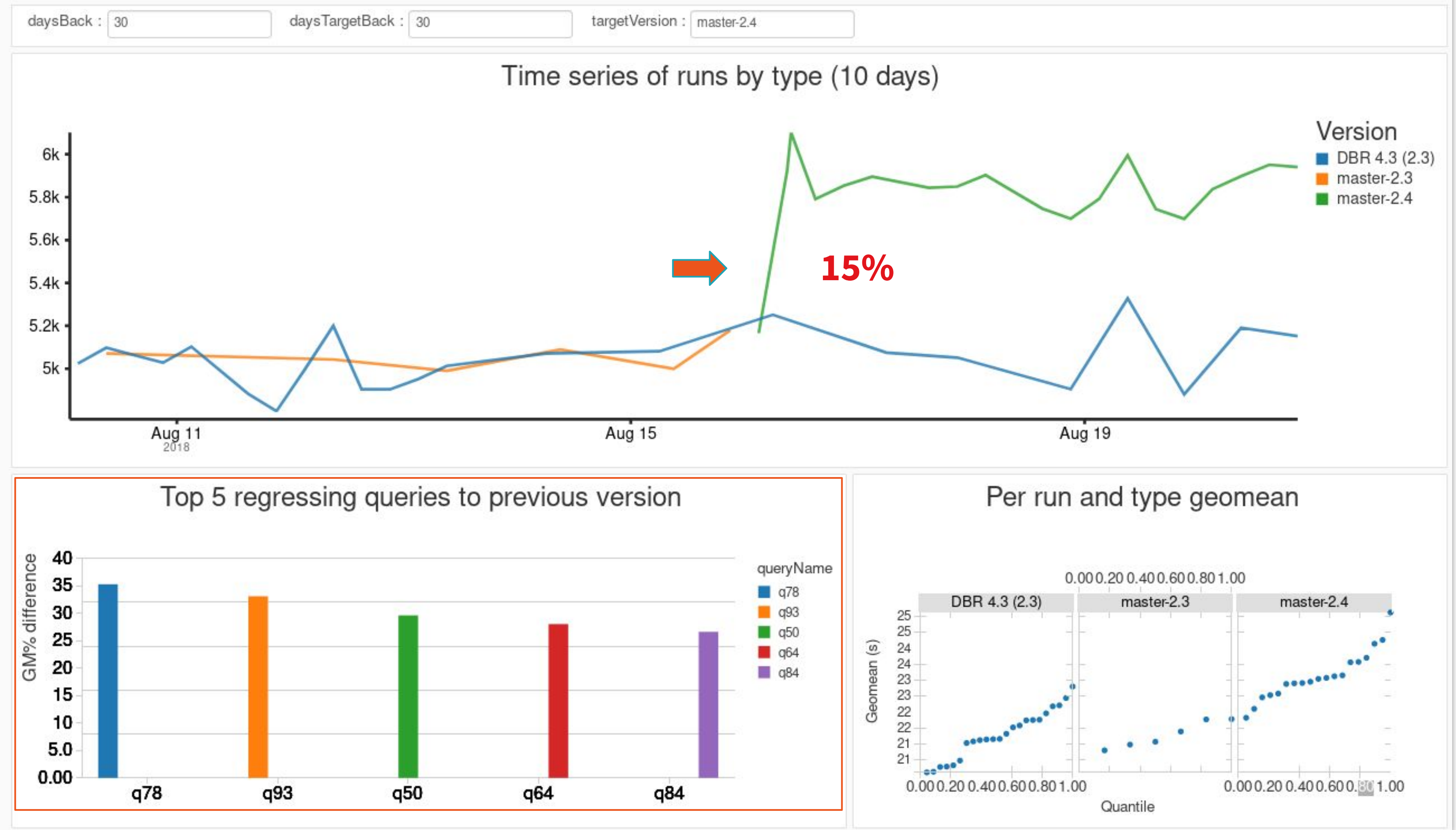
Running from command line.

https://github.com/databricks/spark-sql-perf



http://www.brendangregg.com/flamegraphs.html

DBR 5.0-beta (v2.4) performance ~~tracking~~-- journey



daysBack : 30

daysTargetBack : 30

targetVersion : master-2.4

Time series of runs by type (10 days)

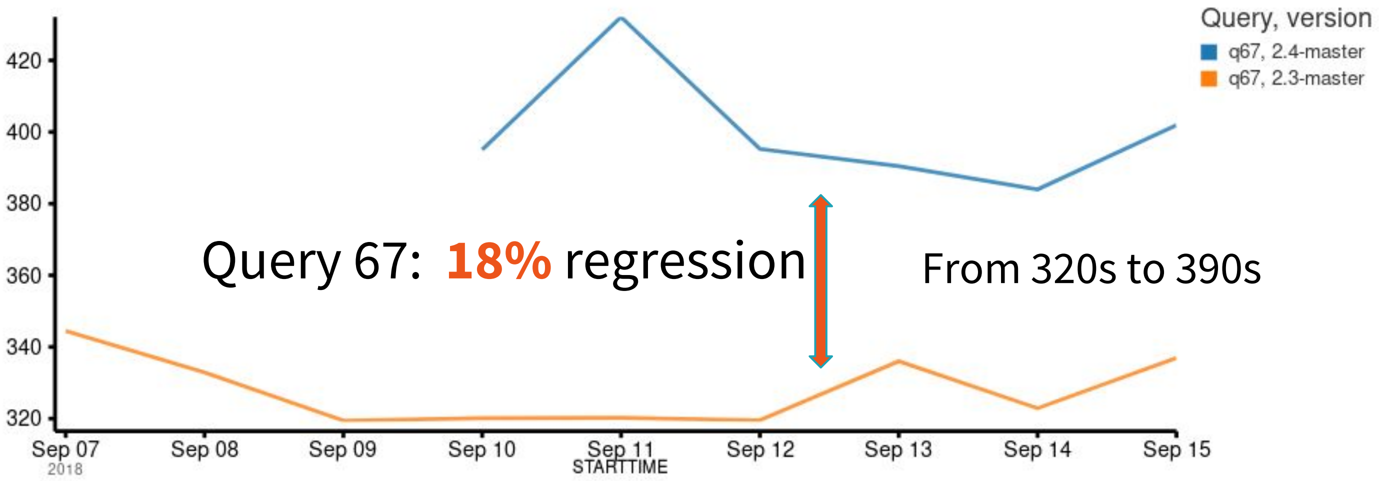
15%

Top 5 regressing queries to previous version

Per run and type geomean

Per query drill-down: 67

First, **scope** and **validate**

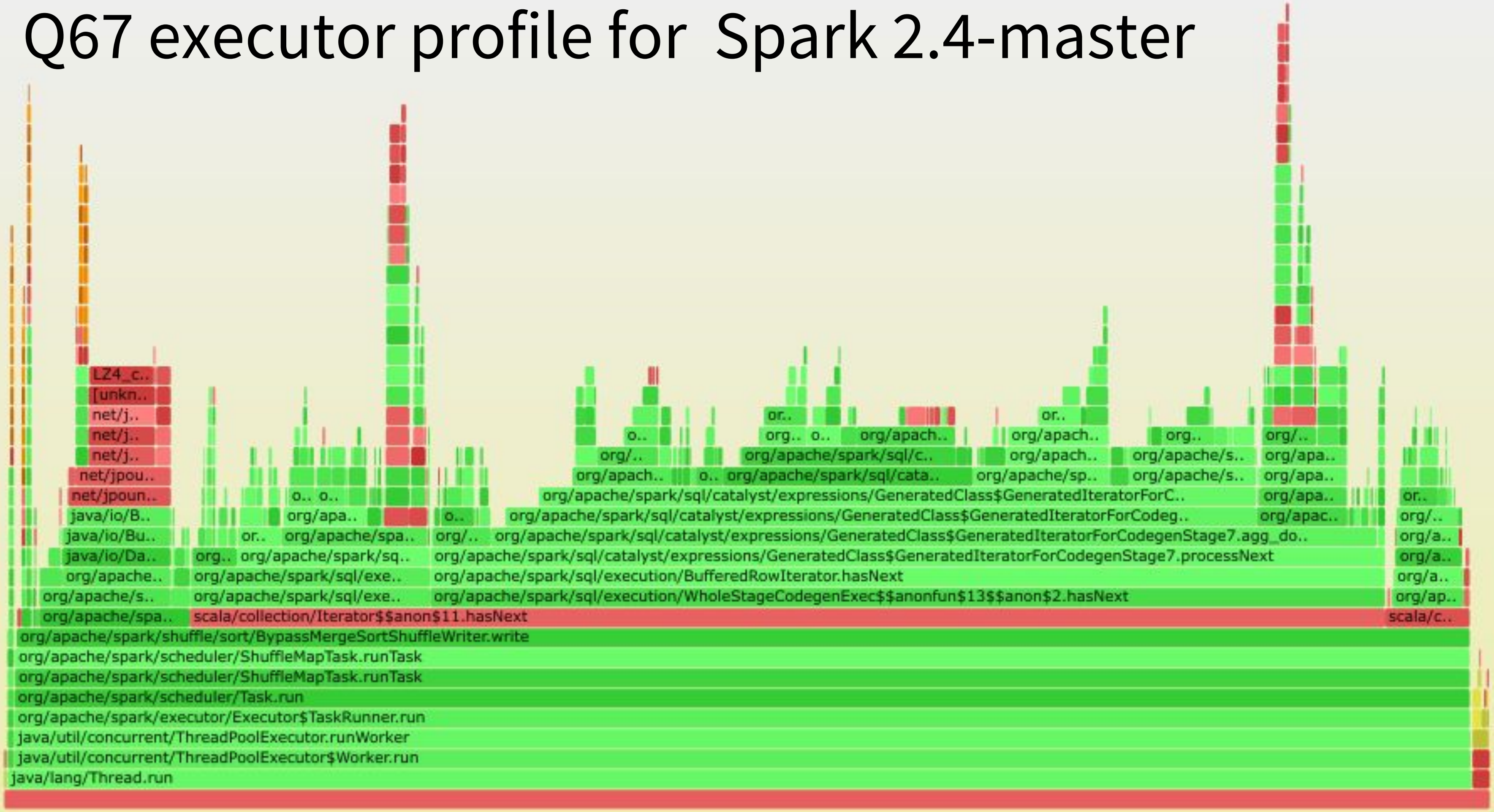


Query 67: **18%** regression

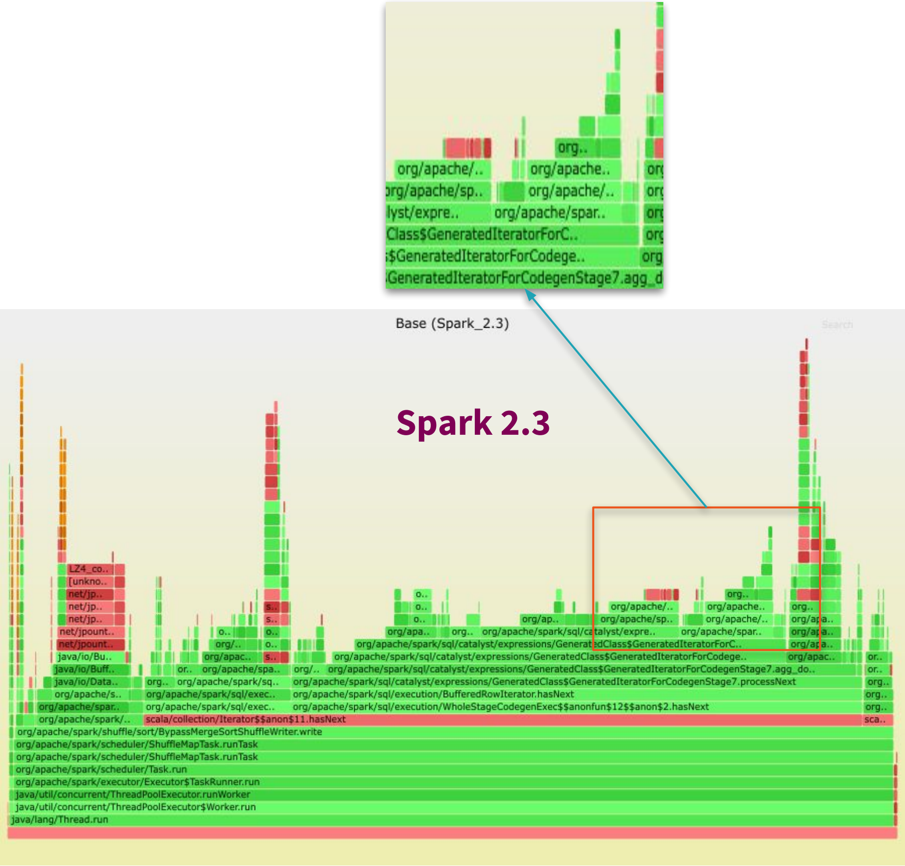
From 320s to 390s

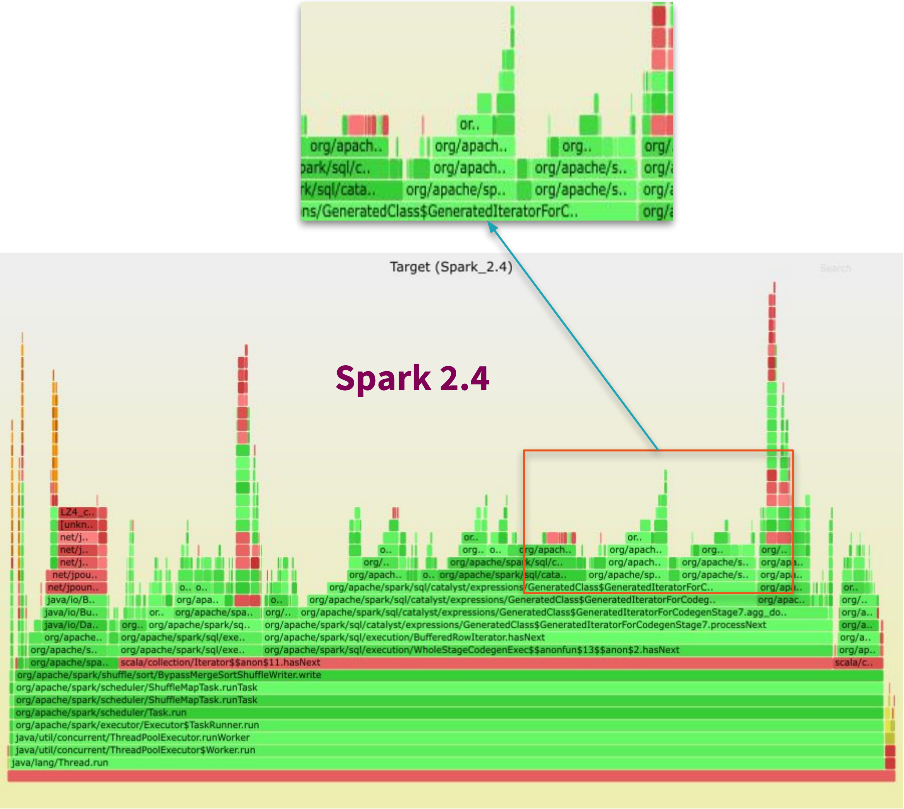
- in 2.4-master (dev) compared
- to 2.3 in DBR 4.3 (prod)

Q67 executor profile for Spark 2.4-master



Side-by-side 2.3 vs 2.4: find the differences



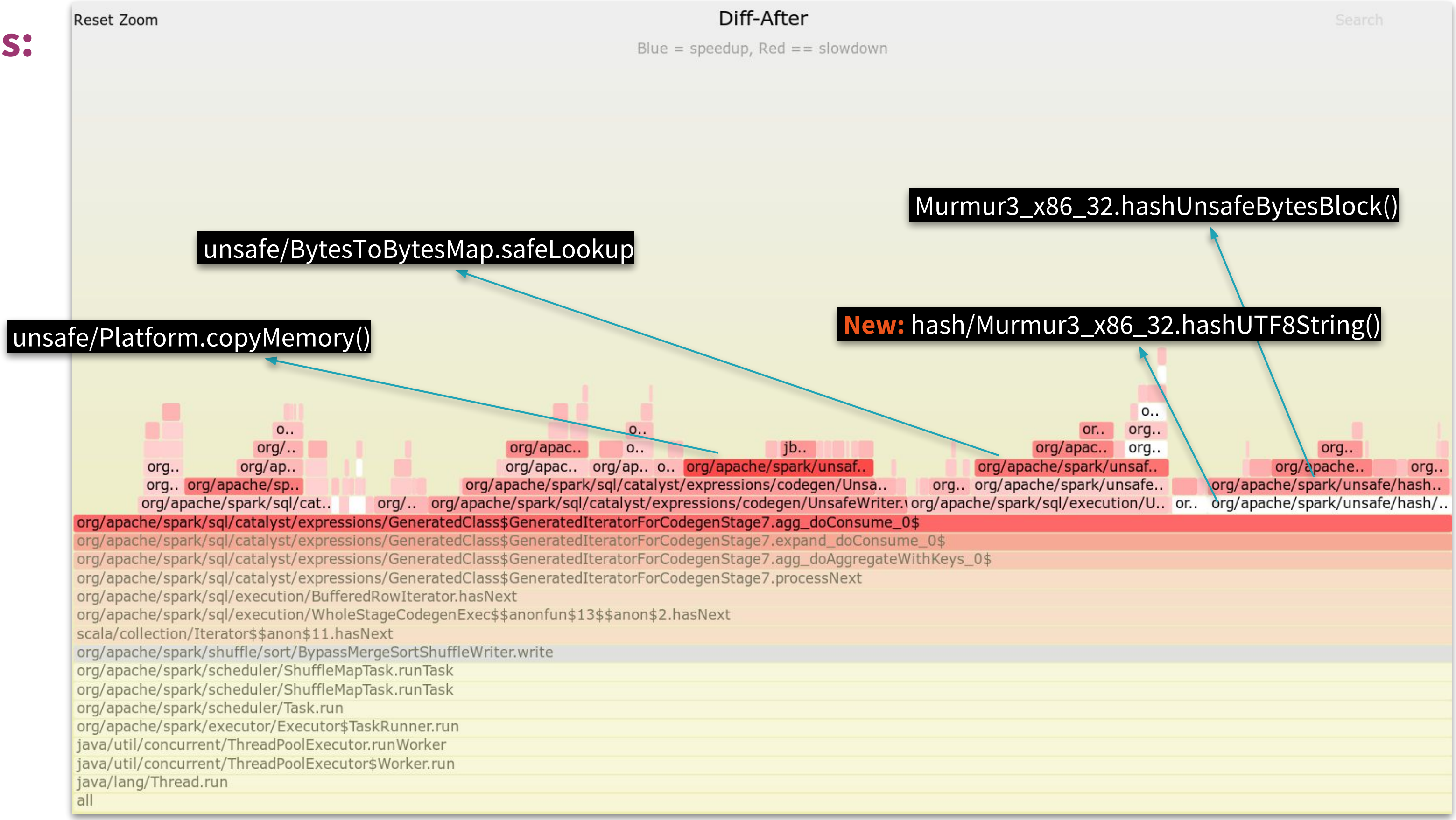


Framegraph diff zoom

Red slower **White** new

Look for hints:

- Mem mgmt
- Hashing
- unsafe



Reset Zoom
Diff-After
Blue = speedup, Red == slowdown
Search
o..
org/..
org..
org/ap..
org..
org/apache/sp..
org/apache/spark/sql/cat..
org/..
org/apac..
o..
org/apac..
org/ap..
o..
jb..
org/apache/spark/unsaf..
org/apache/spark/sql/catalyst/expressions/codegen/Unsa..
org/apache/spark/sql/catalyst/expressions/codegen/UnsafeWriter.
org..
org/apache/spark/sql/execution/U..
or..
org..
org/apac..
org..
org/apache/spark/unsaf..
org/apache/spark/unsafe..
o..
org..
org/apache..
org..
org/apache/spark/unsafe/hash..
or..
org/apache/spark/unsafe/hash/..
org/apache/spark/sql/catalyst/expressions/GeneratedClass$GeneratedIteratorForCodegenStage7.agg_doConsume_0$
org/apache/spark/sql/catalyst/expressions/GeneratedClass$GeneratedIteratorForCodegenStage7.expand_doConsume_0$
org/apache/spark/sql/catalyst/expressions/GeneratedClass$GeneratedIteratorForCodegenStage7.agg_doAggregateWithKeys_0$
org/apache/spark/sql/catalyst/expressions/GeneratedClass$GeneratedIteratorForCodegenStage7.processNext
org/apache/spark/sql/execution/BufferedRowIterator.hasNext
org/apache/spark/sql/execution/WholeStageCodegenExec$$anonfun$13$$anon$2.hasNext
scala/collection/Iterator$$anon$11.hasNext
org/apache/spark/shuffle/sort/BypassMergeSortShuffleWriter.write
org/apache/spark/scheduler/ShuffleMapTask.runTask
org/apache/spark/scheduler/ShuffleMapTask.runTask
org/apache/spark/scheduler/Task.run
org/apache/spark/executor/Executor$TaskRunner.run
java/util/concurrent/ThreadPoolExecutor.runWorker
java/util/concurrent/ThreadPoolExecutor$Worker.run
java/lang/Thread.run
all

Murmur3_x86_32.hashUnsafeBytesBlock()

unsafe/BytesToBytesMap.safeLookup

New: hash/Murmur3_x86_32.hashUTF8String()

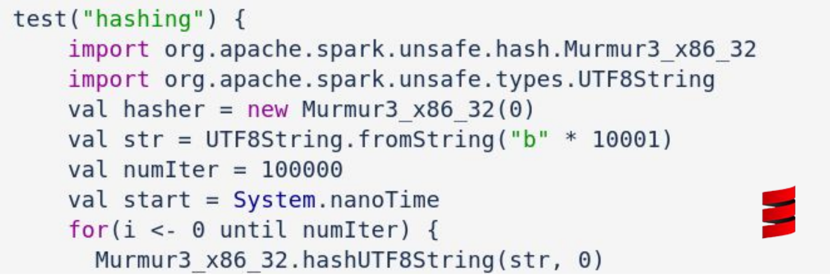
unsafe/Platform.copyMemory()

databricks

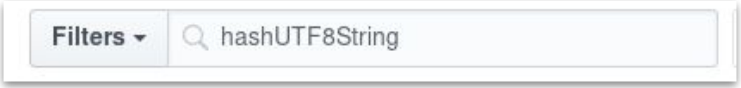
Root-causing

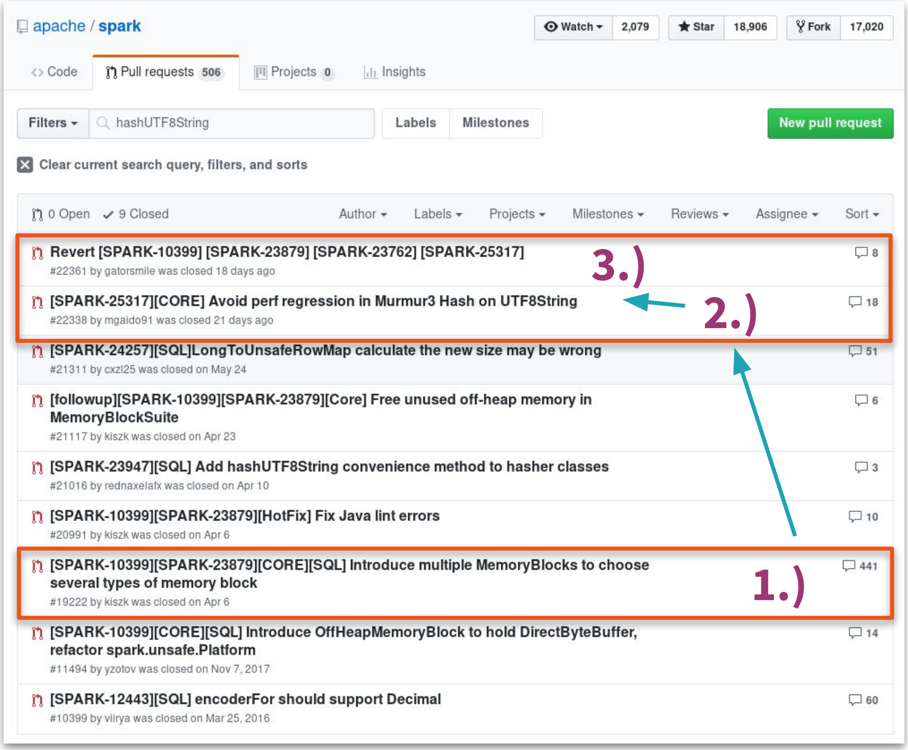
GIT BISECT

Microbenchmark for UTF8String



```
test("hashing") {
    import org.apache.spark.unsafe.hash.Murmur3_x86_32
    import org.apache.spark.unsafe.types.UTF8String
    val hasher = new Murmur3_x86_32(0)
    val str = UTF8String.fromString("b" * 10001)
    val numIter = 100000
    val start = System.nanoTime
    for(i <- 0 until numIter) {
      Murmur3_x86_32.hashUTF8String(str, 0)
```





Revert [SPARK-10399] [SPARK-23879] [SPARK-23762] [SPARK-25317] — 8
#22361 by gatorsmile was closed 18 days ago

[SPARK-25317][CORE] Avoid perf regression in Murmur3 Hash on UTF8String — 18
#22338 by mgaido91 was closed 21 days ago

[SPARK-24257][SQL]LongToUnsafeRowMap calculate the new size may be wrong — 51
#21311 by cxzl25 was closed on May 24

[followup][SPARK-10399][SPARK-23879][Core] Free unused off-heap memory in MemoryBlockSuite — 6
#21117 by kiszk was closed on Apr 23

[SPARK-23947][SQL] Add hashUTF8String convenience method to hasher classes — 3
#21016 by rednaxelafx was closed on Apr 10

[SPARK-10399][SPARK-23879][HotFix] Fix Java lint errors — 10
#20991 by kiszk was closed on Apr 6

[SPARK-10399][SPARK-23879][CORE][SQL] Introduce multiple MemoryBlocks to choose several types of memory block — 441
#19222 by kiszk was closed on Apr 6

[SPARK-10399][CORE][SQL] Introduce OffHeapMemoryBlock to hold DirectByteBuffer, refactor spark.unsafe.Platform — 14
#11494 by yzotov was closed on Nov 7, 2017

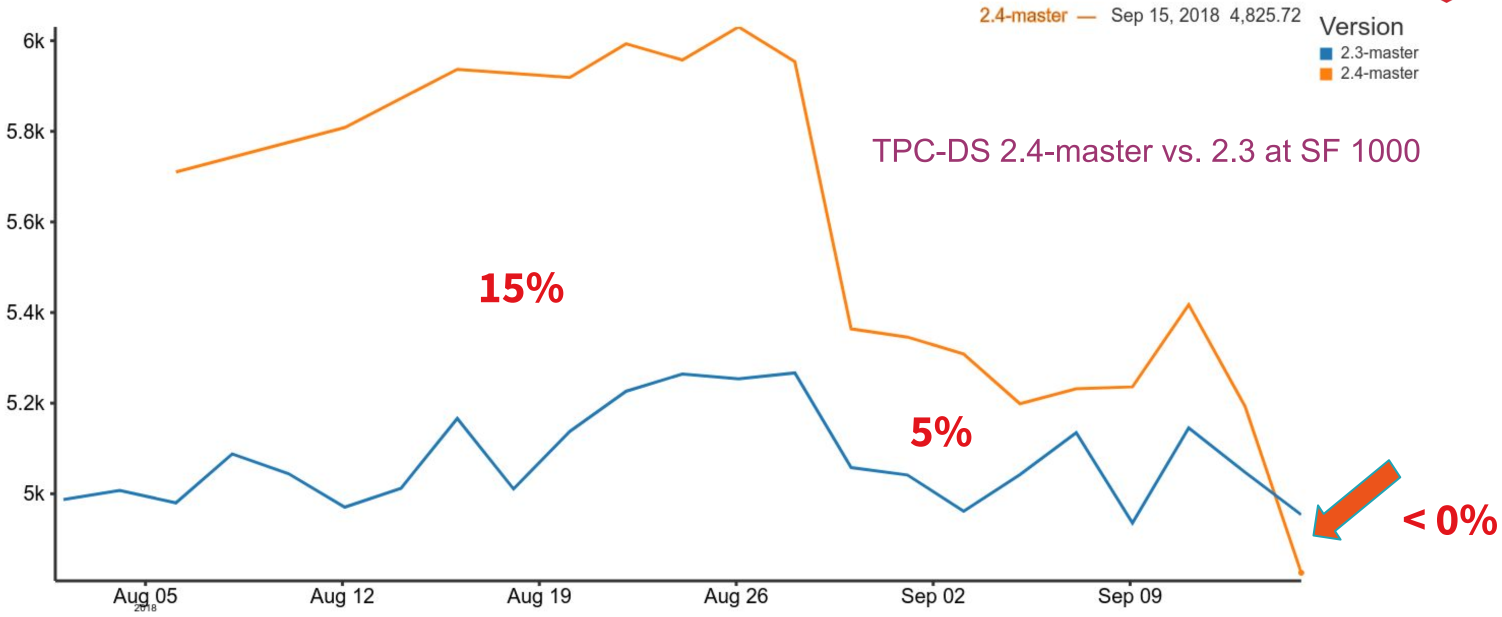
[SPARK-12443][SQL] encoderFor should support Decimal — 60
#10399 by viirya was closed on Mar 25, 2016

Results:

- Spark 2.3: hashUnsafeBytes() -> **40µs**
- Spark 2.4 *hashUnsafeBytesBlock()* -> **140µs**

- also slower *UTF8String.getBytes()*

It is a journey to get a release out



DBR and Spark testing and performance are a **continuous effort**

- **Over a month** effort to bring performance to improving

Conclusion

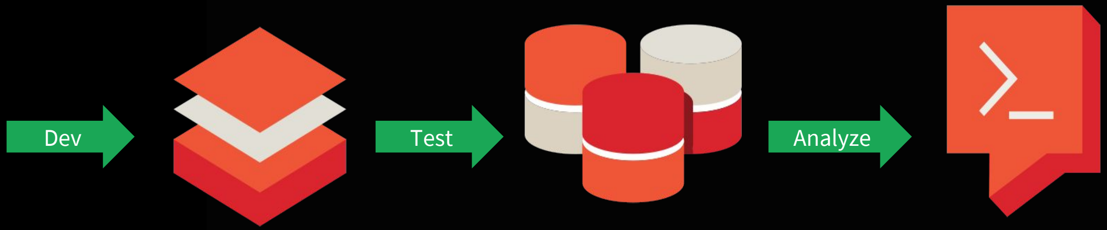
Spark in production is *not just the framework*

Unit and integration testing are not enough

We need Spark specific tools to automate the process

to ensure both correctness and performance

Thanks!



Correctness and Performance of Apache Spark SQL

October 2018

databricks