kontextfrei

A new approach to testable Spark applications

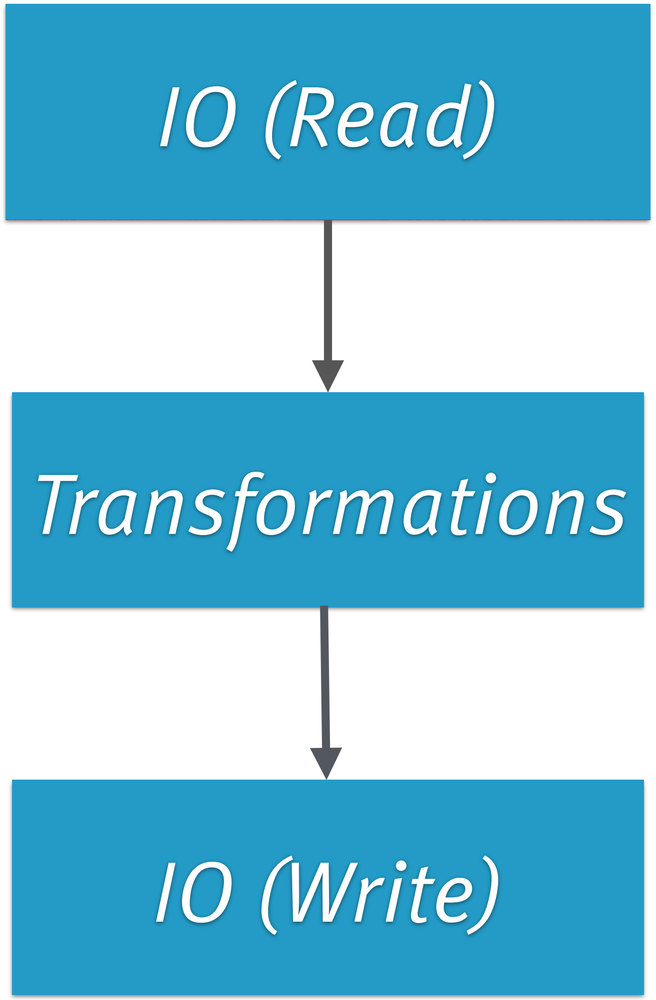
Daniel Westheide / @kaffeecoder

Scalar Conf, 08.04.2017

innoQ

Recap: Spark in a nutshell

- framework for distributed processing of big and fast data
- core abstraction: `RDD[A]`


IO (Read)
Transformations
IO (Write)

IO actions

```
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD

val sparkContext = new SparkContext("local[1]", "test-app")

val rawRepoStarredEvts =
  sparkContext.textFile("test-data/repo_starred.csv")
```

Transformations

```
import org.apache.spark.rdd.RDD

def usersByPopularity(repoStarredEvts: RDD[RepoStarred]): RDD[(String, Long)] =
  repoStarredEvts
    .map(evt => evt.owner -> 1L)
    .reduceByKey(_ + _)
    .sortBy(_._2, ascending = false)
```

Testing Spark apps

- Does it succeed in a realistic environment?
- Does it succeed with real-world data sets?
- Is it fast enough with real-world data sets?
- **Is the business logic correct?**

Property-based testing

- good fit for testing pure functions
- specify properties that must hold true
- tested with randomly generated input data

Spark +

property-based testing =

♡

?

Testing business logic

- fast feedback loop
- write test ~› fail ~› implement ~› succeed
- `sbt ~testQuick`

Testing RDDs

- always requires a `SparkContext`
- property-based testing: functions are tested with lots of different inputs

Twitter

Person 1

Secure https://twitter.com

Apps truhe Other Bookmarks

Home Notifications Messages Search Twitter Tweet

Daniel Westheide
@kaffeecoder

TWEETS	FOLLOWING	FOLLOWERS
3,391	403	2,232

Germany Trends · Change

#sgesvw
4,986 Tweets

#ECHO2017
@luebken is Tweeting about this

#Syrien

#letsdance
2,301 Tweets

#SignOfTheTimes
1.14M Tweets

Menschenmenge
7,578 Tweets

terrorverdacht
1,801 Tweets

Assad
639K Tweets

Menschen Leben Tanzen Welt

What's happening?

William Gibson Retweeted

Zeke Miller @ZekeJMiller · 5h
A family affair

Ivanka Trump @IvankaTrump
Very proud of Arabella and Joseph for their performance in honor of President Xi Jinping and Madame Peng Liyuan's official visit to the US!

14 10 23

Eberhard Wolff Retweeted

Frank Düsterbeck @fduesterbeck · 8h
Replying to @ewolff
Nee is klar ... und Wolff heißt rückwärts Fflow 😜

Translate from German

1

You might like

Helena Edelson and Thomas Lockney liked

Greg Wild @GregJWild · 15h
Dear neanderthals of the world: whether you're using tomahawks, trucks or sarin gas, kindly accelerate your regression into the sea.

Who to follow · Refresh · View all

WIRED @WIRED Follow

Paul Phillips @contrarivariant Follow

James Roper @jroper Follow

Find friends

Coping strategies

- global `SparkContext` for all tests
- unit testing only the functions you pass to **RDD** operators
- being lazy about testing

- *kontextfrei – adjective | kon·text–frei:* of, relating to, or being a grammar or language based on rules that describe a change in a string without reference to elements not in the string; also: being such a rule[1]
- other meanings: the state of being liberated from the chains of the SparkContext

1: see https://www.merriam-webster.com/dictionary/context-free

kontextfrei

```
resolvers +=
  "dwestheide" at “https://dl.bintray.com/dwestheide/maven",

libraryDependencies ++= Seq(
  "com.danielwestheide" %% "kontextfrei-core-spark-2.1.0" % "0.5.0",
  "com.danielwestheide" %% "kontextfrei-scalatest-spark-2.1.0" % "0.5.0"
)
```

https://github.com/dwestheide/kontextfrei

https://dwestheide.github.io/kontextfrei/


ABSTRACT
OVER ALL THE THINGS!
memegenerator.net

kontextfrei

- abstracts over **RDD**
- business logic and test properties without **RDD** dependency
- execute on **RDD** or local Scala collections

Design goal

- as close to Spark as possible
  - execution model
  - user API
- extensible

Typeclasses FTW!

```
trait DCollectionOps[DCollection[_]] {
  def map[A: ClassTag, B: ClassTag](as: DCollection[A])(
      f: A => B): DCollection[B]
}
```

Business logic

```
import com.danielwestheide.kontextfrei.DCollectionOps

class JobLogic[DCollection[_]: DCollectionOps] {

 import com.danielwestheide.kontextfrei.syntax.Imports._

 def usersByPopularity(repoStarredEvts: DCollection[RepoStarred])
   : DCollection[(String, Long)] = {
   repoStarredEvts
     .map(evt => evt.owner -> 1L)
     .reduceByKey(_ + _)
     .sortBy(_._2, ascending = false)
 }

}
```

```scala
class RDDOps extends DCollectionOps[RDD] {

 override final def map[A: ClassTag, B: ClassTag](as: RDD[A])(
     f: A => B): RDD[B] = as.map(f)
}

trait RDDOpsSupport {

 implicit def rddCollectionOps(
     implicit sparkContext: SparkContext): DCollectionOps[RDD] =
   new RDDOps(sparkContext)
}
```

Glue code

```
import com.danielwestheide.kontextfrei.rdd.RDDOpsSupport
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD

object Main extends App with RDDOpsSupport {
 implicit val sparkContext = new SparkContext("local[1]", "test-app")
 try {
   val logic = new JobLogic[RDD]
   val repoStarredEvts = logic
     .parseRepoStarredEvents(
       sparkContext.textFile("test-data/repo_starred.csv"))
   val usersByPopularity = logic.usersByPopularity(repoStarredEvts)
   logic
     .toCsv(usersByPopularity)
     .saveAsTextFile("target/users_by_popularity.csv")
 } finally {
   sparkContext.stop()
 }
}
```

Test support

- base trait for tests (highly optional)
- generic `Gen[DCollection[A]]` and `Arbitrary[DCollection[A]]` instances
- generic `Collecting` instance

App-specific base spec

```
trait BaseSpec[DColl[_]]
    extends KontextfreiSpec[DColl]
    with DCollectionGen
    with CollectingInstances
    with Generators
    with PropSpecLike
    with GeneratorDrivenPropertyChecks
    with MustMatchers
```

Test code

```
import com.danielwestheide.kontextfrei.syntax.Imports._

trait UsersByPopularityProperties[DColl[_]] extends BaseSpec[DColl] {

 def logic: JobLogic[DColl]

 property("Total counts correspond to number of events") {
   forAll { starredEvents: DColl[RepoStarred] =>
     val result =
       logic.usersByPopularity(starredEvents).collect().toList
     result.map(_._2).sum mustEqual starredEvents.count()
   }
 }

}
```

Testing for correctness

```
class UsersByPopularitySpec
    extends UnitSpec
    with UsersByPopularityProperties[Stream] {
 override val logic = new JobLogic[Stream]
}
```

Verify that it works ;)

```
import org.apache.spark.rdd.RDD

class UsersByPopularityIntegrationSpec
    extends IntegrationSpec
    with UsersByPopularityProperties[RDD] {
 override val logic = new JobLogic[RDD]
}
```

Workflow

- local development:
  - `> sbt ~testQuick`
  - very fast feedback loop
- CI server:
  - `> sbt test it:test`
  - slower, catching more potential runtime errors

Demo time

Alternative design

- Interpreter pattern
- Describe computation, run it with a Spark or local executor
- implemented by Apache Crunch: Hadoop pipeline, Spark, and in-memory pipelines

Shortcomings

- only supports **RDD**
- sometimes in Spark, business logic cannot be cleanly isolated
- not all **RDD** operators implemented (yet)
- no support for broadcast variables or accumulators
- API exposes advanced Scala features

Summary

- Spark doesn’t really support unit testing
- kontextfrei restores the fast feedback loop
- early stage, but successfully used in production
- looking for contributors

Thank you for your attention!

Twitter: @kaffeecoder

Website: danielwestheide.com