Speaker Notes: Scala for Apache Spark

Markus Dale, medale@asymmetrik.com

Jan 2019

Setup

- Open Spark API:
  https://spark.apache.org/docs/latest/api/scala/index.html
- Open Scala API: https://www.scala-lang.org/api/2.11.8/#package
- Open Java API String: https://docs.oracle.com/javase/8/docs/api/

Intro, Slides And Code

- Bio:
  - mostly Java, big data with Hadoop
  - big data with Spark, Databricks, Scala
  - Now Asymmetrik - Scala, Spark, Elasticsearch, Akka...
  - Data Engineer
- Slides: https://github.com/medale/scala-spark/blob/master/presentation/ScalaSpark.pdf
- Scala Spark Code Examples: https://github.com/medale/scala-spark
- Also https://github.com/medale/spark-mail

Goals

- Intro to Scala (from Java) to leverage Apache Spark with Scala API
- sbt - build tool
- spark-testing framework for integration testing

Why Scala for Spark?

- Data Engineer - scalable ecosystem of Java/Scala-based tools
- less boilerplate/less typing (Ted Malaska (three big data books on O'Reilly): 50% less than Java)
- strong typing, elegant multi-paradigm language (functional and OO)
- all code runs in executor JVM - no callouts to local Python shell for UDFs/UDAFs
- *spark-shell* is Scala-based
- immutability for parallel/distributed computing
- Baltimore Scala meetup

Java to Scala - Java Main

- semicolons
- get/set JavaBeans convention
- explicit constructor
- static main method

Scala Main One

- Look Ma - no semicolons
- package structure - match directory structure
- Match file name/class name
- object vs. class
    - object - Java static methods, singleton
- no public (default)
- def - method/function declaration
- type declared after variable name
- parameterized type: `Array[String]`
- return type, body (last entry gets returned)
- val - immutable, var - mutable
- class constructor (args none vs. val vs. var)
    - Java get/set: `import scala.reflect.BeanProperty`
    - `@BeanProperty var firstName`

HelloSparkWorld - expression-oriented

- expressions returns value (vs. statements)
  - Array (any indexed sequence) accessor args(index)
  - type of lines is inferred as lowest-common denominator if/else block
  - In this case: `Seq[String]`

Scala Type Hierarchy

- Main division - AnyVal vs. AnyRef (unified through Any)
- AnyRef is like Object in Java
    - Null is a subclass of all reference classes
- AnyVal - Java primitives
    - Unit - `val u = () //u: Unit = () 0-tuple`
    - Value class: `class Wrapper(val underlying: Int) extends AnyVal`
- Universal trait:
    - trait that extends Any
    - only has defs as members
- Nothing is subclass of everything (throwing exception returns Nothing)

HelloSparkWorld - String, StringOps, implicits

- Triple quoted string can include special chars like newline, double-quote
- .stripMargin - by default uses pipe | removes all chars in front of pipe
- String - where does stripMargin method come from? (next slide)

Java API - String

- java.lang.String - does not have stripMargin method

Scala Predef API - implicit conversions

- `implicit def augmentString(x: String): StringOps`
- Also: `implicit def booleanArrayOps(xs: Array[Boolean]): ArrayOps[Boolean]`

Scala StringOps API - stripMargin

- stripMargin method
- Also useful: head, tail, map, filter, sliding (ngrams), permutations

HelloSparkWorld - accessing Java API/libraries

- import aliasing: `import java.util.{List => JavaList}`
- `readLinesFromFile` uses nio.Paths/Files (could use any 3rd party Java library, e.g. Apache Commons IO)
- `import scala.collection.JavaConverters._` - implicit conversions
  - underscore like Java * - import all methods from JavaConverters
  - DecorateAsScala/DecorateAsJava/AsScala/AsScala
- last line `lines` gets returned from method (return type Seq[String])

wordCountLocal: map higher-order function w/named function

- Scala 1 JVM processing - Seq trait (Array implements)
- higher-order functions - input is a function (or returns function)
- map - iterate over seq, one input, one output element
    - Immutability: underlying seq is not updated, returns new seq!
- map - with named function
    - can define function within a function (also imports)
    - if small function - pollute namespace, harder to read

wordCountLocal: map higher-order function w/ function literal

- syntactic sugar 1 - type is inferred
- placeholder syntax - if parameter is only used once
    - for two param function, first , *second*

map function

- new collection
- same number of elements
- use function to transform each element (doesn't have to be same type)

flatMap function

- map + flatten - map function must return collection (GenTraversableOnce)
- flatten - takes each element of result collection and appends to output in order
    - only flattens outermost collection!

wordCountLocal: flatMap and filter

- flatMap - one input element - GenTraversableOnce (Seq-like), 0 or more
- filter - keep elements that test true

Scala Seq trait API

- Show filter, flatMap higher-order functions

wordCountLocal: foldLeft

- create empty map String to Int, default value 0
- two args: 1 arg - same as return type (here map)
- second arg: function of (map, current word)
    - look up value for word and increment by 1 (default 0)
    - if not default - NoSuchElementException
    - create new map with updated key - newValue pair (replace existing key)

wordCountLocal: mkString

- mkString
- string interpolation - s“...”

HelloSparkWorld - SparkSession

- object - main method entry point
- SparkSession Scala API
    - scaladocs
    - object
    - class
    - .builder method (don't need empty parentheses - mutator method with ())
- Builder class - fluent interface/method chaining
    - getOrCreate
    - Run from shell or batch spark-submit

SparkSession Scala API

- Showed SparkSession scaladocs

HelloSparkWorld - RDD map, flatMap, filter

- process at scale!
- parallelize from driver to executor
- Immutable RDD - map, flatMap, filter transformations!
    - action like collect, take, write causes execution

HelloSparkWorld - RDD of tuples - PairRDDFunctions

- groupBy - expensive shuffle operation
- map to 2 tuple - implicit conversion to PairRDDFunctions
  - import org.a.s.rdd.RDD
  - object RDD - `implicit def rddToPairRDDFunctions(rdd: RDD[(K, V)])`
- PairRDDFunctions - reduceByKey - function with two arguments
  - local combine step, then shuffle (hashPartitioner)
- *transformations (lazy)* executed by *action* collect()!
  - to local driver - memory!!!

RDD object API

- implicit conversions of RDD of type x - to, for example:
  - (K,V) pairs - rddToPairRDDFunctions
  - Double/Numeric - double/numericRDDToDoubleRDDFunctions

HelloSparkDatasetWorld - Scala case class

- immutable data structure
- default constructor parameters are `val`
- generates boiler plate code, singleton object
  - apply method - constructor without new
  - also unapply for pattern matching
- Scala: Lots of “syntactic sugar” - less typing, compiler translates
  - access elements in indexed collection: coll(index) - coll.apply(index)

HelloSparkDatasetWorld - javap Person.class

- javap disassembler (package names removed)
- implements Product (abstract algebraic type), Serializable
    - Serializable important for Spark shuffle!
    - Product (productPrefix = classname)
        - productArity
        - productElement(int)
        - productIterator
- static apply factory method/unapply for matching
- copy method
    - Scala - named parameters (can create new object with one changed param)
    - `p.copy(age=43)`
- equals, hashCode, toString
    - toString - classname(field1,field2...)
- also curried, tupled methods

HelloSparkDatasetWorld - Encoder

- Encoder: Spark manages objects in memory (minimize garbage collection)
    - import implicits for Encoders for Scala primitives, String, Product..
- select returns DataFrame (a Dataset[Row])
- where with Column - $ convert string to Column
    - less than method on Column
    - === for column equality (== for Scala equality (object and primitives)!!)

org.apache.spark.sql.functions._

- Column-based manipulation of column content
  - time
  - string
  - math
- aggregate functions with RelationalGroupedDataset.agg (after groupBy)

Integration Testing - make code testable as you write

- Tests basic logic (not scalability - partitioning, OOM errors, key skew)
- Separate out dataset creation (from file, directory etc.)
- Method returns testable value (or write to local file dir, test)

Integration Testing - ScalaTest with Spark Testing Base

- ScalaTest - FunSuite (xTest), FunSpec more BDD specs
- Spark Testing Base - trait DatasetSuiteBase
    - `spark` SparkSession
    - Dataset equality, approximate equality...
- Test boundaries etc.

Integration Testing - assertExpectedCountForCutoff

- Use DatasetSuiteBase spark
- refactor common testing
- Matchers - should equal

sbt - “build tool for Scala, Java and more”

- Scala, Java and mixed, C++
- build.sbt - modules/libraries used
- version.sbt - project version
- project/build.properties - sbt version to use
    - plugins.sbt - e.g. assembly
    - Dependencies.scala
- subproject standard directory layout

sbt - Dependencies.scala

- Libraries and version
- % vs. %% (spark-core_2.11)

sbt - build.sbt

- analytics and common subprojects
- Scope: ThisBuild common settings for subprojects
- assemblyPluginSettings - fat jar file (assembly)
- dependsOn - other subproject
- Integration test settings - it:test
- subproject name for published artifacts
- library dependencies from Dependencies.scala

sbt tasks

- start interactive sbt shell
- compile, test (root project)
- it:test in analytics subproject
- console - with dependencies
- run - finds all Main classes (can invoke with args)
- assembly - fat jar
    - under target/scala-2.11
- publish, publishLocal (also publishTo, Mavenstyle)

And now for something completely different: Colon Cancer

Resources

- Dean Wampler, Lightbend - great Scala learning/reference
    - Safari Books online - ACM
- Jacek Laskowski - updated content through Spark 2.4 (latest version)
- Bill Venners/Artima - ScalaTest
- Holden Karau - Spark Testing Base, speaker, Spark maintainer/PMC, now at Google
- sbt reference manual