27 maja, 2019

PRZETWARZANIE DANYCH Z UŻYCIEM TANDEMU SPARK I SCALA

Technologie Programistyczne, Magda i Piotr Miara

O CZYM BĘDZIE TA PREZENTACJA?

Scala

Składnia

Programowanie funkcyjne

Zastosowania

Apache Spark

Elementy składowe

Podstawowe typy danych

Porady wydajnościowe

Databricks

Funkcjonalność

Demo

SCALA

JĘZYK PROGRAMOWANIA
OGÓLNEGO PRZEZNACZENIA

HISTORIA

Autor - Martin Odersky (EPFL, Szwajcaria)

Opublikowana w 2004 roku

Obecna wersja 2.12.8

CECHY

Programowanie obiektowe i funkcyjne

Interoperacyjność z Javą

"Lepsza Java"

Skalowalność

HELLO WORLD

```
object HelloScala {
  def main(args: Array[String]): Unit = {
    println("Hello World!")
  }
}

object HelloScala extends App {
  println("Hello World!")
}
```

PRZYPISANIE

```
val a = 5
val b: Int = 5

var c = 10
c = 20

def d = a
def e = (x: Int) => x + 1
```

KLASA

```
class Student {
  var name: String = _
  var grades: List[Int] = Nil
}

val myUser = new User
myUser.name = "John"
myUser.grades = List(2,2,3,2)
```

GETTERY I SETTERY

```
class Student {
  private var priv_name: String = _
  var grades: List[Int] = Nil
  def name = priv_name
  def name_=(name: String) = {
    validate(name)
    priv_name = name
  }
}
```

CASE CLASS

```
case class Student(name: String, grades: List[Int])

val nowak = Student("Nowak", List(2,2,2))

println(nowak)
// Student(Nowak,List(2, 2, 2))

val nowak_2 = Student("Nowak", List(2,2,2))
nowak == nowak_2
// true
```

PATTERN MATCHING

```
import scala.util.Random

val x: Int = Random.nextInt(10)

x match {
  case 0 => "zero"
  case 1 => "one"
  case 2 => "two"
  case _ => "other"
}
```

PATTERN MATCHING

```
abstract class Animal
case class Cat(name: String, livesLeft: Int) extends Animal
case class Dog(name: String, favouriteToy: String) extends Animal

def sayHello(pet: Animal): String = {
  pet match {
    case Cat(name, _) => s"Miau, miau, I'm $name"
    case Dog(name, toy) => s"I'm $name and I like $toy"
  }
}
```

PROGRAMOWANIE FUNKCYJNE

- CZYSTE FUNKCJE (PURE FUNCTIONS)
- NIEZMIENNOŚĆ (IMMUTABILITY)
- FUNKCJE PIERWSZEJ KLASY
- FUNKCJE WYŻSZEGO RZĘDU

F. WYŻSZEGO RZĘDU

```
val numbers = (1 to 100).toList

numbers.map(x => x * 2)
numbers.map(_ * 2)

numbers.filter(_ % 2 == 0).reduce(_ + _)
```

F. WYŻSZEGO RZĘDU

```
def add_n(n: Int): (Int) => Int = {
  (x: Int) => n + x
}

def add_10 = add_n(10)

println(add_10(5))
// 15
```

Scala wciąż jest dość mało popularnym językiem (30. miejsce w rankingu Tiobe), ale wkrótce może się to zmienić.
Istnieje kilka frameworków (np. Play), które ostatnio mocno zyskują na popularności.
O wartości Scali świadczą bardzo znane narzędzia napisane z jej użyciem: Apache Kafka, Akka oraz **Apache Spark**.

APACHE SPARK

PLATFORMA PROGRAMISTYCZNA DLA
OBLICZEŃ ROZPROSZONYCH

HISTORIA

Projekt badawczy na UC Berkeley, AMPlab

Opublikowany w 2010

Przekazany Apache Fundation w 2013

CECHY

Ponad 1000 osób wspierających rozwój projektu

Największy otwartoźródłowy projekt w dziedzinie przetwarzania danych

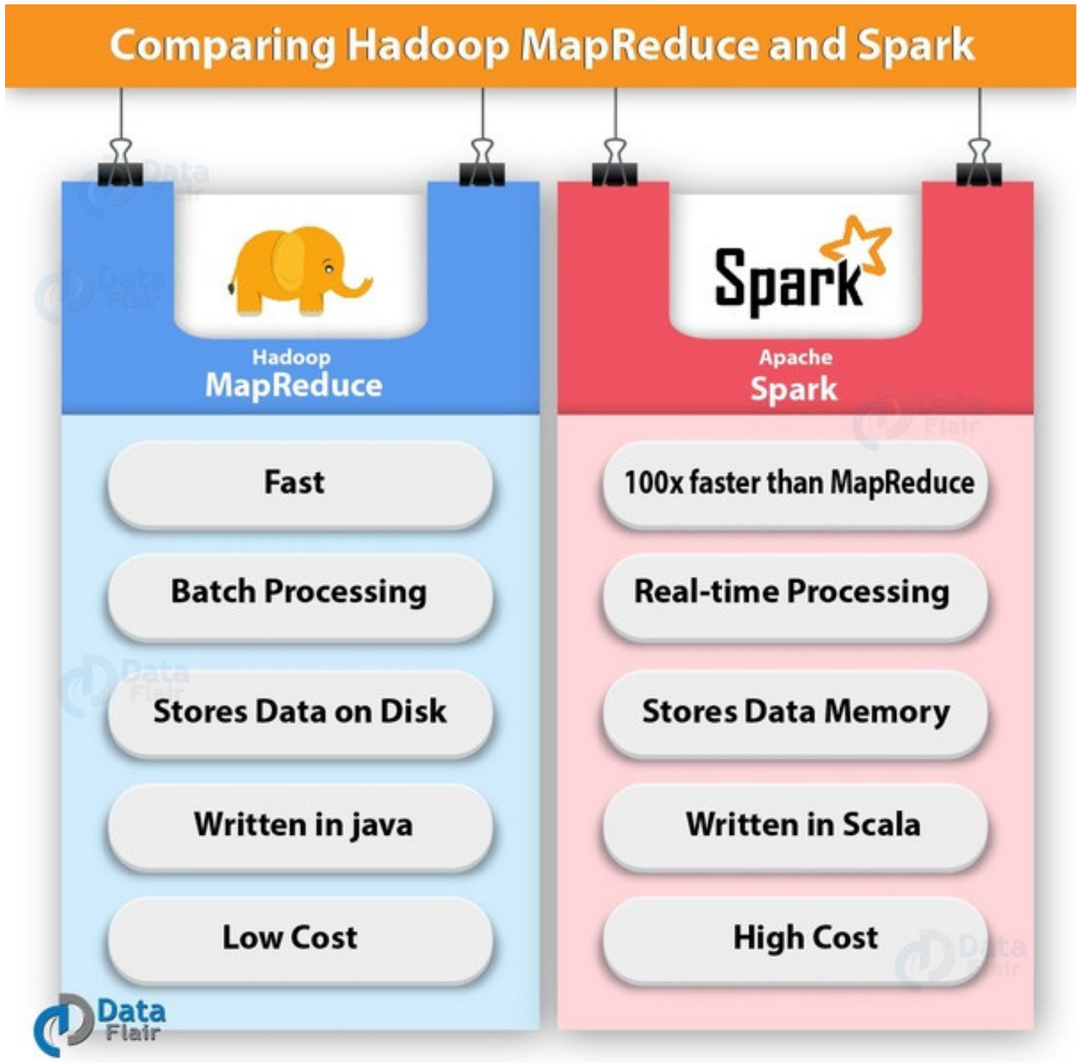
Integracja z wieloma językami programowania

GŁÓWNA ZALETA SPARKA

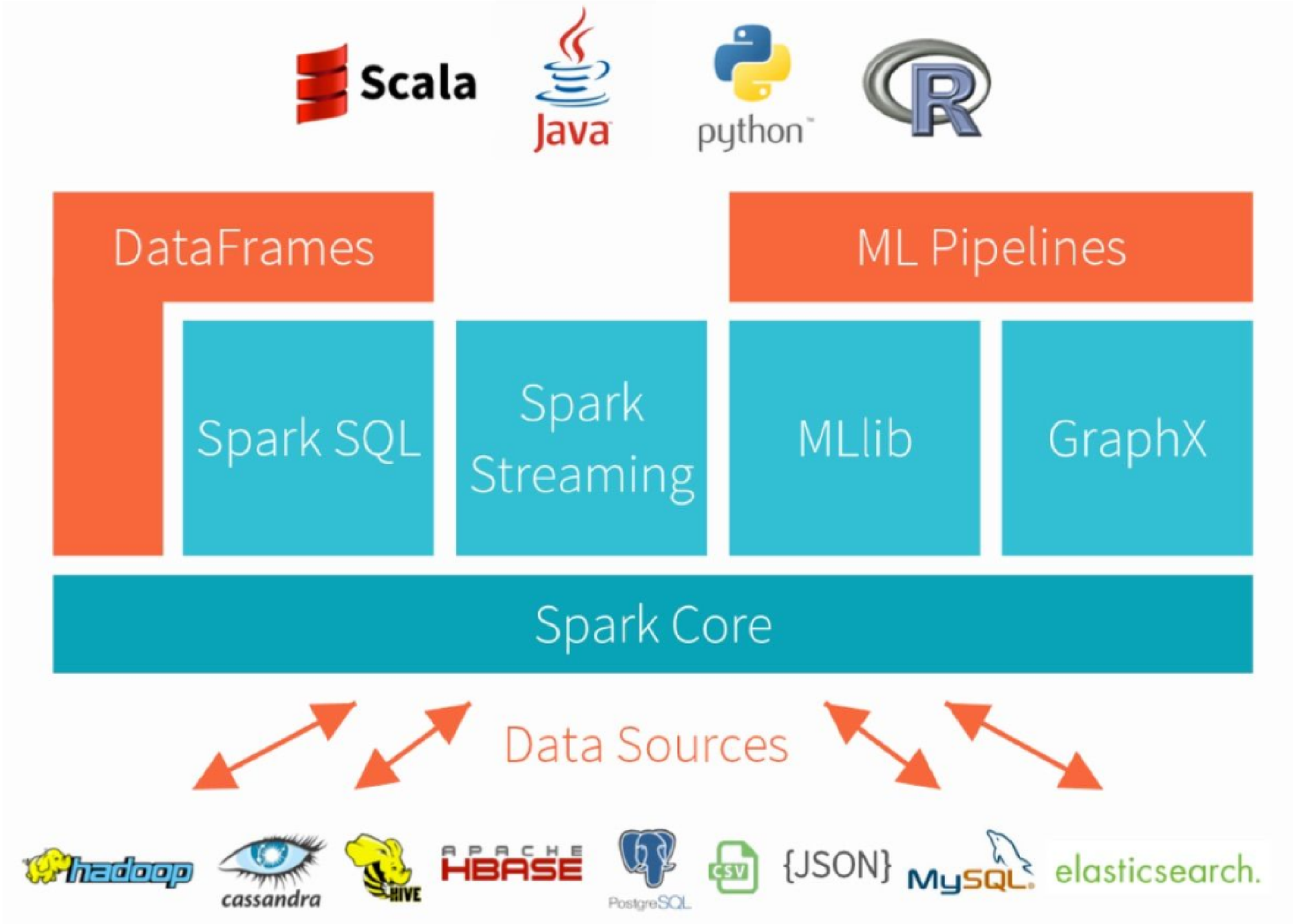
Pierwsza rzecz jaką chwalą się twórcy Spark jest jego szybkość. Wysoką wydajność osiąga dzięki przechowywaniu wyników pośrednich w pamięci RAM, krótkim czasie uruchamiania zadań oraz automatycznej optymalizacji zapytań.

Hadoop MapReduce

Spark często jest porównywany do swojego poprzednika - Hadoop MapReduce, którego bije na głowę jeśli chodzi o szybkość.



Główne komponenty Sparka



PODSTAWOWE TYPY DANYCH

RDD

Resilient Distributed Datasets, podstawowa abstrakcja Sparka, dane nieustrukturyzowane

DATAFRAME

Analogia do tabel w bazach danych, dane mają określoną strukturą, podzielone na kolumny

DATASET

Dataframe z typami, zorientowany obiektowo, sprawdzanie błędów na etapie kompilacji

RDD

```
val simpleData = sc.parallelize(Array(1,2,3))

val lines = sc.textFile("data.txt")
val lineLengths = lines.map(s => s.length)
val totalLength = lineLengths.reduce(_ + _)
```

DATAFRAME

```
val df = spark.read.json("people.json")
df.show()
// +----+-------+
// | age|   name|
// +----+-------+
// |null|Michael|
// |  30|   Andy|
// |  19| Justin|
// +----+-------+
df.filter($"age" > 21).select("name")
```

DATASET

```
case class Person(name: String, age: Long)
val df = spark.read.json("people.json").as[Person]
df.show()
// +----+-------+
// | age|   name|
// +----+-------+
// |null|Michael|
// |  30|   Andy|
// |  19| Justin|
// +----+-------+
df.filter(_.age > 21).map(_.name)
```

LOCAL

Najprostszy sposób, eksperymentowanie i nauka

STANDALONE

Model rozproszony. Spark zarządza dostępnymi zasobami. Dowolne źródło danych.

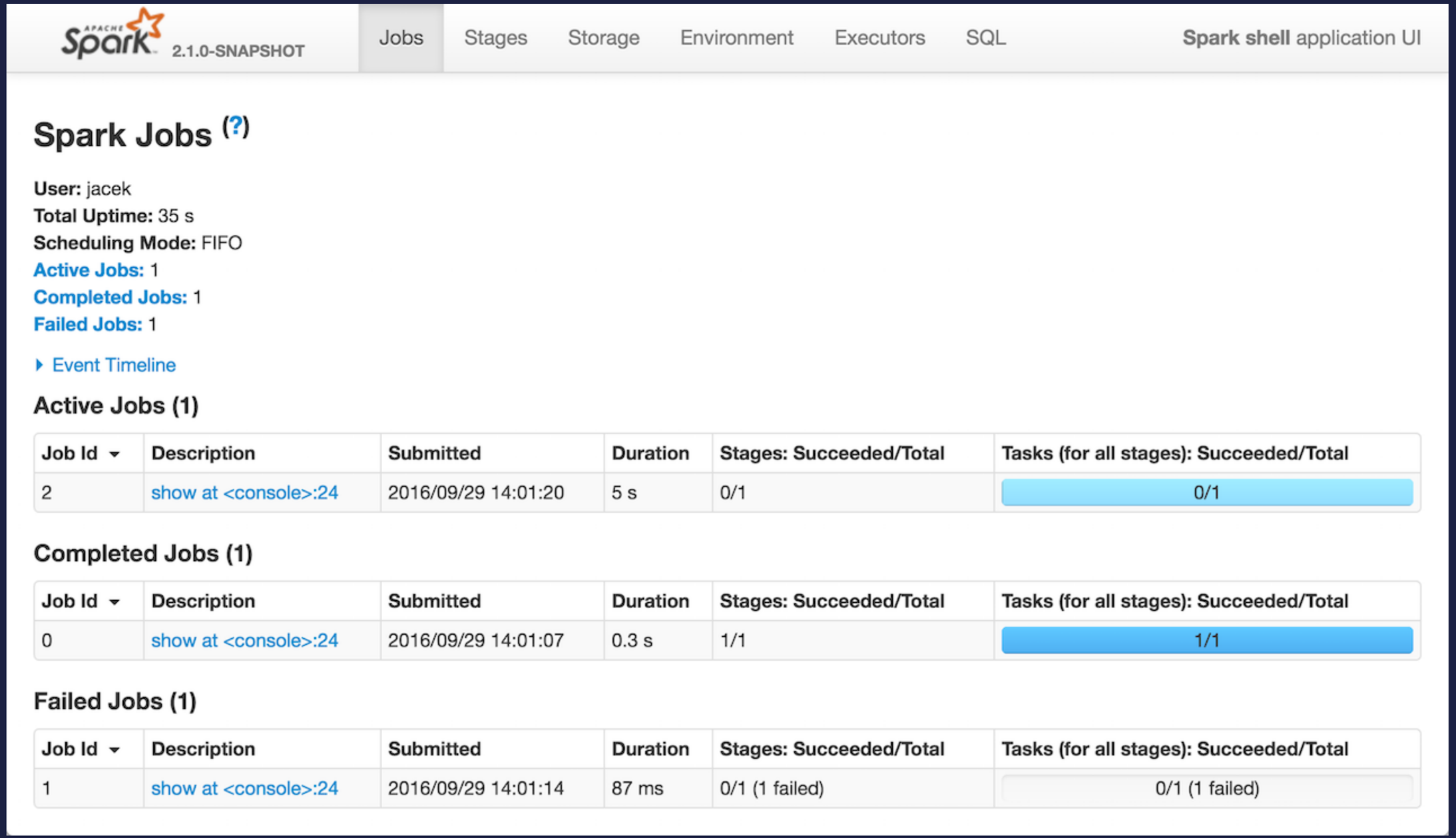
YARN

Model rozproszony, przystosowany do HDFS. YARN używany jest również przez Hadoop MapReduce

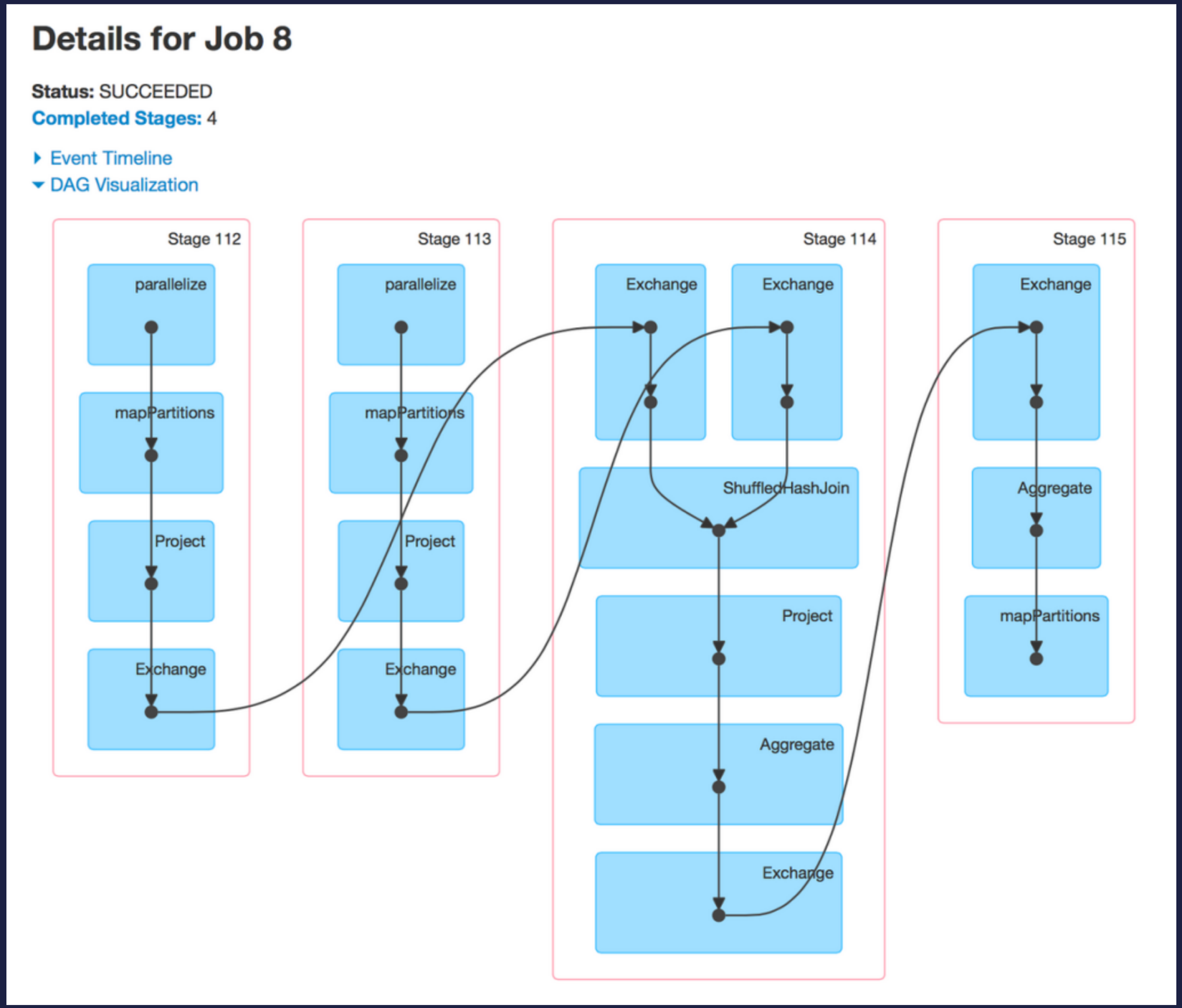
MESOS

Podobne rozwiązanie do YARN, z tym że Apache Mesos odpowiada za przydzielanie zasobów

Spark UI


Spark 2.1.0-SNAPSHOT
Jobs
Stages
Storage
Environment
Executors
SQL
Spark shell application UI
Spark Jobs (?)
User: jacek
Total Uptime: 35 s
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 1
Failed Jobs: 1
Event Timeline
Active Jobs (1)
Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total
2 | show at <console>:24 | 2016/09/29 14:01:20 | 5 s | 0/1 | 0/1
Completed Jobs (1)
Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total
0 | show at <console>:24 | 2016/09/29 14:01:07 | 0.3 s | 1/1 | 1/1
Failed Jobs (1)
Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total
1 | show at <console>:24 | 2016/09/29 14:01:14 | 87 ms | 0/1 (1 failed) | 0/1 (1 failed)

DAG


Details for Job 8
Status: SUCCEEDED
Completed Stages: 4
Event Timeline
DAG Visualization
Stage 112
parallelize
mapPartitions
Project
Exchange
Stage 113
parallelize
mapPartitions
Project
Exchange
Stage 114
Exchange
Exchange
ShuffledHashJoin
Project
Aggregate
Exchange
Stage 115
Exchange
Aggregate
mapPartitions

PORADY WYDAJNOŚCIOWE

Zbiór porad na podstawie wykładu "Advanced Apache Spark Training" wygłoszonego na Spark Summit 2015.

- sprawdź poziom zrównoleglenia poprzez spojrzenie na liczbę partycji; liczba partycji ~ poziom zrównoleglenia
- przyjrzyj się różnym sposobom cache'owania, np. cache'owanie z serializacją oszczędza pamięć, ale jest wolniejsze
- używaj dysku do cache'owania tylko jeśli obliczenia są naprawdę kosztowne; czasem bardziej opłacalne jest ponowne przeliczenie
- ustaw liczbę core'ów 2 lub 3 razy większą niż liczba rdzeni logicznych

SCALA

NATYWNY JĘZYK SPARKA

PYTHON

NAKŁADKA PYSPARK

Scala
Język programowania

Python
Język programowania

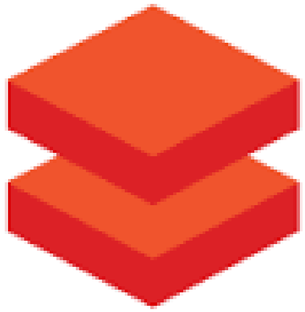
+ Dodaj porównanie

Cały świat | Ostatnie 5 lat | Wszystko | Wyszukiwarka Google

Zainteresowanie w ujęciu czasowym

DATABRICKS

USŁUGA ANALITYCZNA OPARTA NA APACHE SPARK



databricks

HISTORIA

Twórcy Sparka

Firma powstała w 2013

CECHY

Platforma opierająca się na Sparku

Ułatwia zarządzanie klastrami

Pozwala na łatwą współpracę

Dostępna na AWS i Azure


Azure Databricks


Home


Workspace


Recents


Data


Clusters


Jobs


Search


Azure Databricks

Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Import & Explore Data

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a Blank Notebook

Create a notebook to start querying, visualizing, and modeling your data.

Common Tasks


New Notebook


Upload Data


Create Table


New Cluster


New Job


New MLflow Experiment New


Import Library


Read Documentation

Recents

Recent files appear here as you work.

Documentation


Databricks Guide


Python, R, Scala, SQL


Importing Data

load_tables (Python)

Cmd 1

```
1 oddzialy = spark.table("oddzialy")
2 paragony = spark.table("paragony")
3 hierarchia_materialow = spark.table("hierarchia_materialow")
4 materialy = spark.table("materialy")
```

▸ oddzialy: pyspark.sql.dataframe.DataFrame = [ODDZIAL: string, NAZWA_3: string ... 41 more fields]
▸ paragony: pyspark.sql.dataframe.DataFrame = [ODDZIAL: string, MATERIAL: long ... 45 more fields]

Command took 0.20 seconds -- by miara.piotr@zabka.pl at 22.05.2019, 16:58:54 on ADB-AA-GR1-P-WE

Cmd 2

```
1 # some data transformations
```

Shift+Enter to run shortcuts

DZIĘKUJEMY ZA UWAGĘ

PRZYDATNE LINKI

https://docs.scala-lang.org/

https://spark.apache.org/

https://youtube.com/watch?v=7ooZ4S7Ay6Y

https://databricks.com/