Apache Spark

podstawy architektury i działania

Filip Wójcik, Senior Data Scientist

filip.wojcik@outlook.com, filip.wojcik@ue.wroc.pl

http://maddatascientist.eu/dla-studenta

Agenda

1. Wprowadzenie do technologii Apache Spark
2. Podstawowe założenia i architektura
3. Struktury danych – RDD, grafy obliczeniowe i odroczone wykonanie

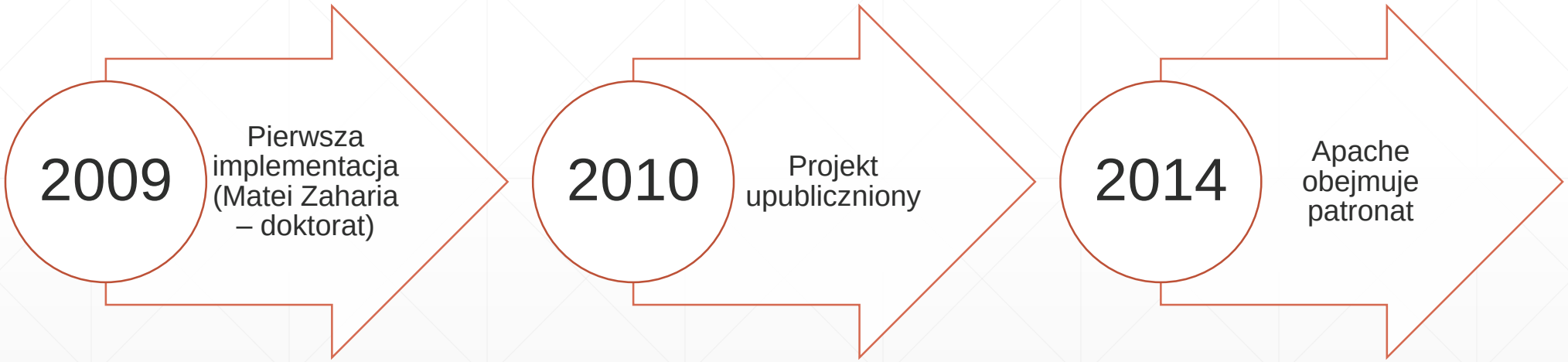
Wprowadzenie do technologii Apache Spark

Informacje o projekcie

Projekt Apache Spark

- Przeznaczony do przetwarzania dużych zbiorów danych W PAMIĘCI OPERACYJNEJ
- Następca klasycznej technologii tzw. Map-Reduce wprowadzonego przez Google
- Projekt open-source, niekomercyjny
- Zaimplementowany natywnie w języku Scala w środowisku JVM
- Posiada interfejsy API (wrappery) w innych językach:
  - Python
  - Java
  - R

Projekt Apache Spark



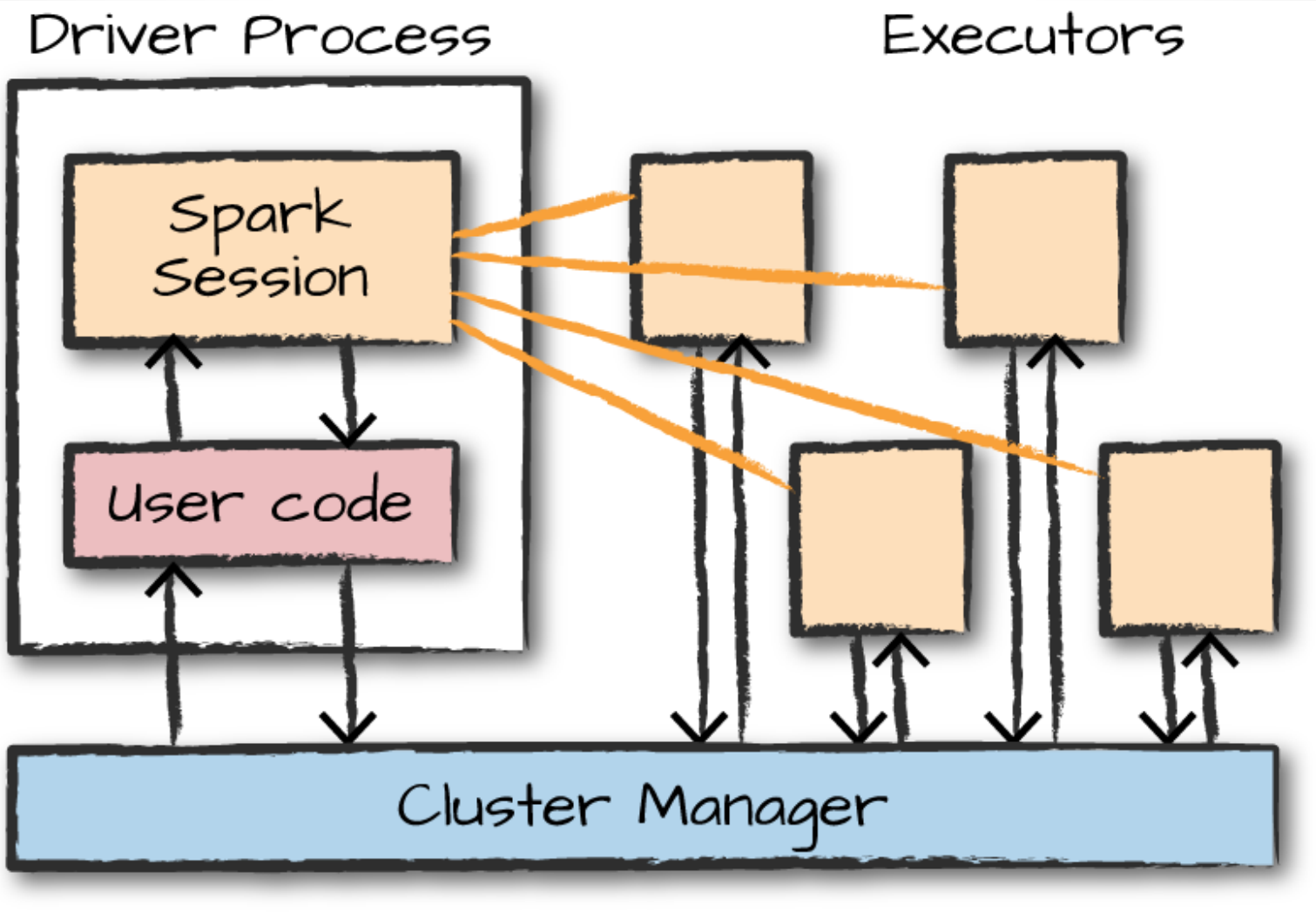
Projekt Apache Spark

- Łatwość przetwarzania zbiorów danych – proste API
- Przetwarzanie w pamięci
- Różne struktury danych – RDD, ramki danych, strumienie danych
- Wiele bibliotek analitycznych do różnych zadań– mllib, graphX
- Wiele języków

- Przetwarzanie wymienne – dysk i pamięć
- Jeden algorytm – map reduce
- API tylko w Javie
- Główne przeznaczenie – przetwarzanie danych

Podstawowe założenia

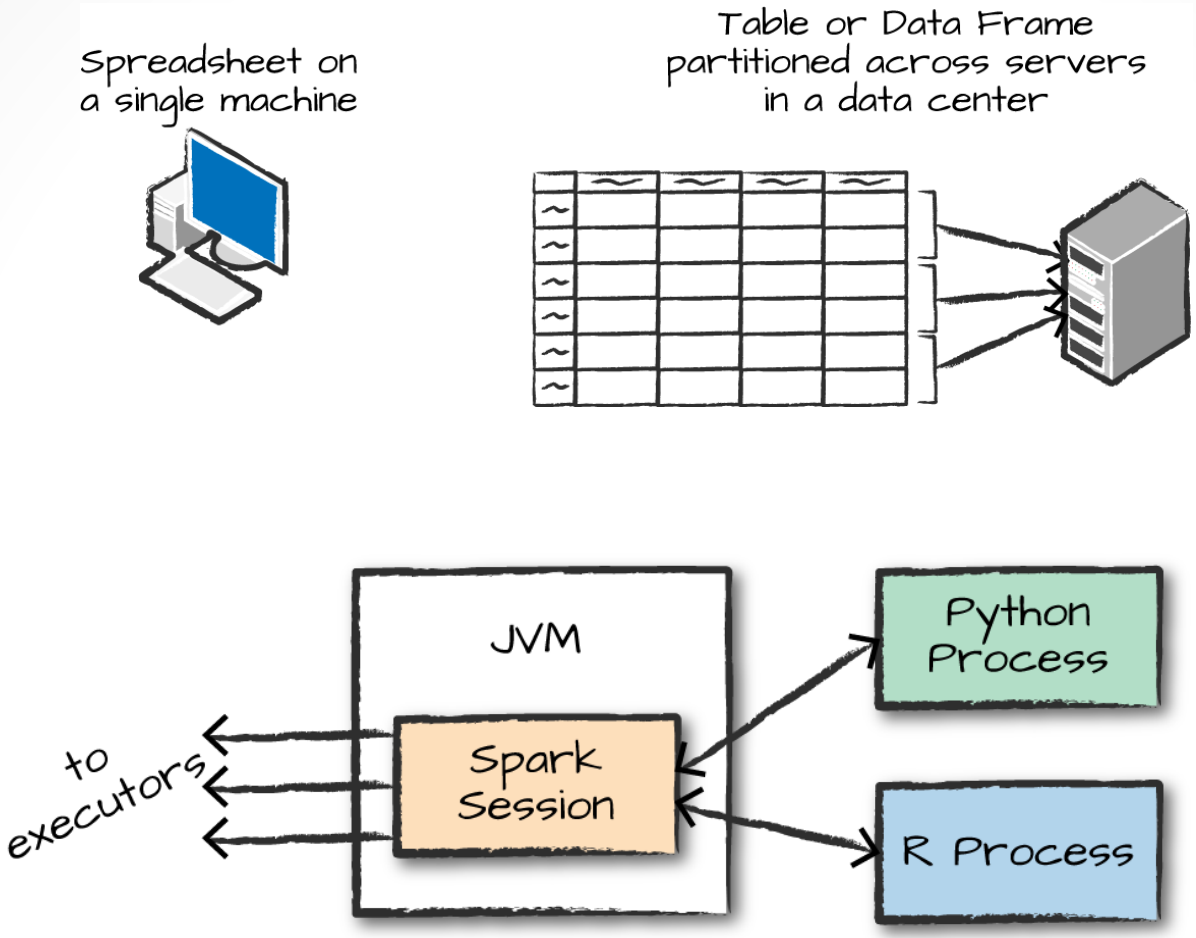
Koncepcja działania Apache Spark



Chambers, Bill, and Matei Zaharia. 2018. *Spark: the definitive guide : big data processing made simple*.

Architektura Sparka

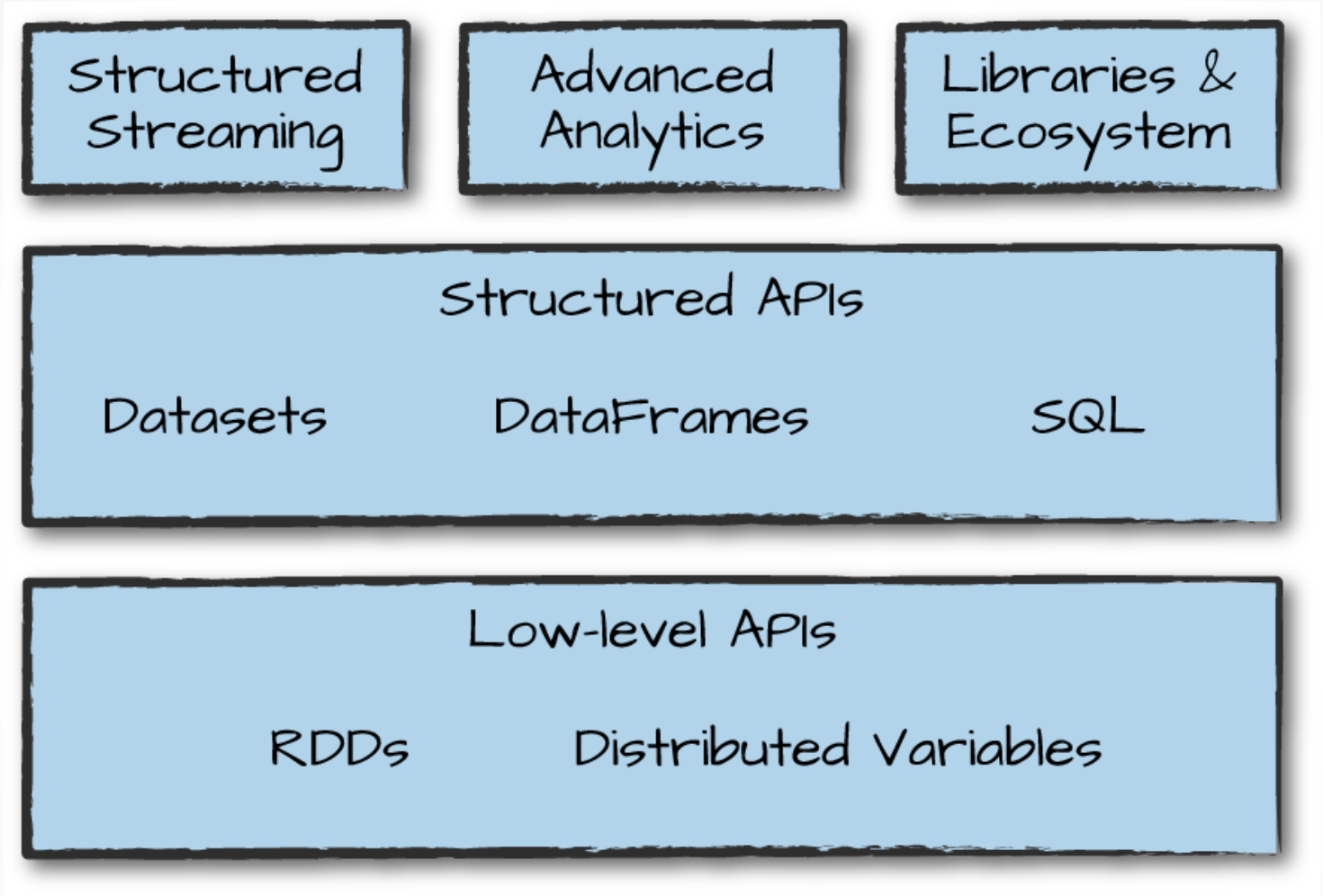
- Architektura **rozproszonego klastra obliczeniowego**
- **Driver – program sterujący** nadzoruje i koordynuje wykonanie operacji
- **Executor** – obecny na maszynach końcowych – **przelicza zadania**



Chambers, Bill, and Matei Zaharia. 2018. *Spark: the definitive guide : big data processing made simple*.

Architektura Sparka

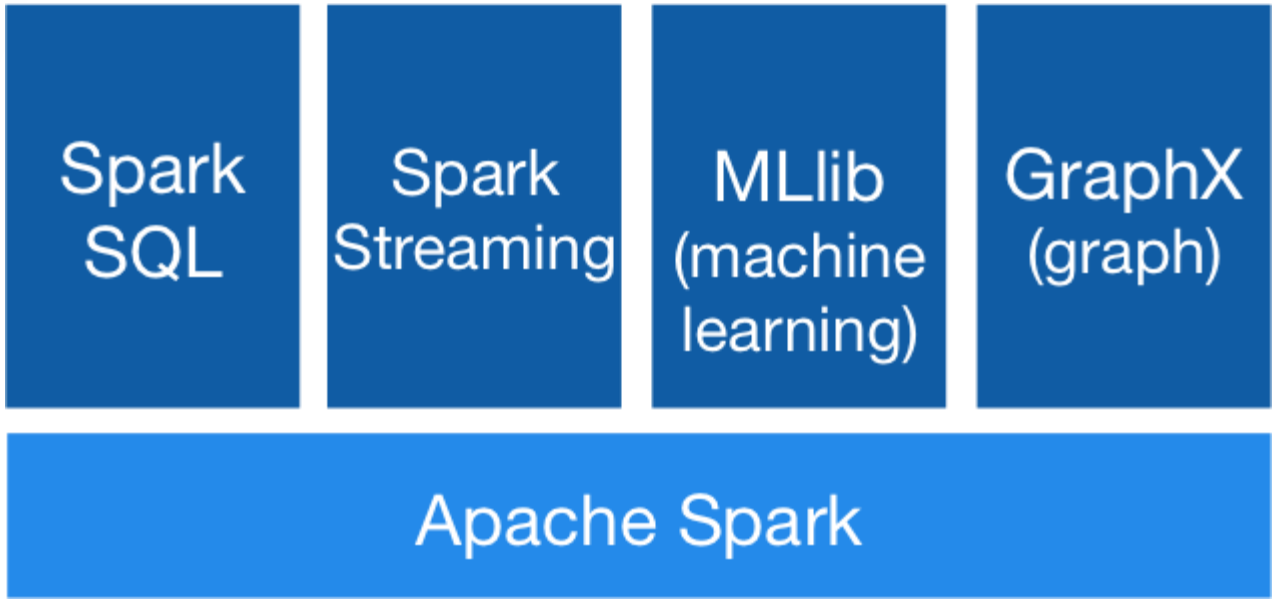
- **Kod użytkownika** zostaje wysłany do tzw. **Sesji Sparka *(Spark Session)***, która komunikuje się ze sterownikiem **(*Driver*)**
- **Wykonanie jest delegowane na klaster**



Chambers, Bill, and Matei Zaharia. 2018. *Spark: the definitive guide : big data processing made simple*.

Hierarchia narzędzi API

- Spark posiada wiele narzędzi różniących się stopniem skomplikowania i złożonością
- Najtrudniejsze są narzędzia niskopoziomowe (podstawowa obsługa RDD)
- Powyżej znajdują się narzędzia do analizy danych uporządkowanych
- Jeszcze wyżej – narzędzia uczenia maszynowego



Ekosystem Sparka

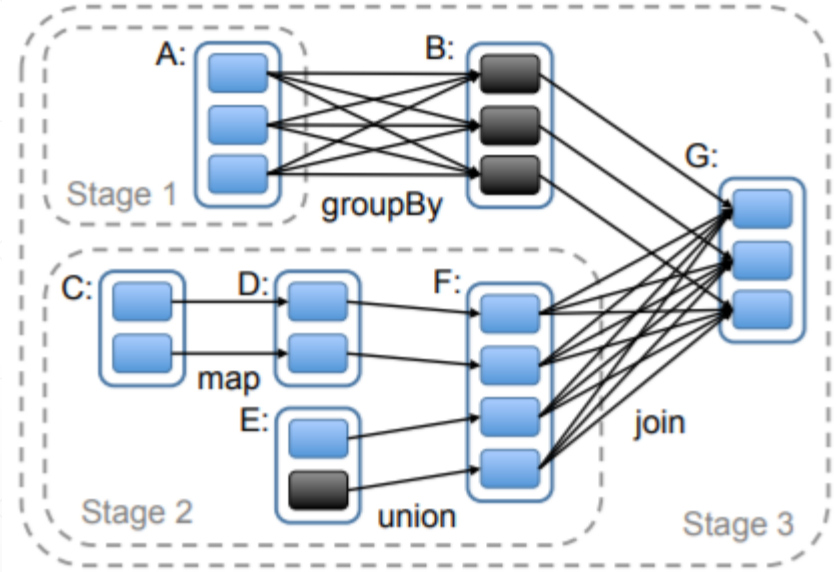
- Apache Spark – podstawa ekosystemu
- Spark SQL – przetwarzanie danych ustrukturyzowanych (tabele)
- Spark Streaming – źródła danych napływające w sposób ciągły (szyny danych)
- MlLib – uczenie maszynowe i sztuczna inteligencja
- GrapX – grafowa baza danych

Struktury danych - RDD

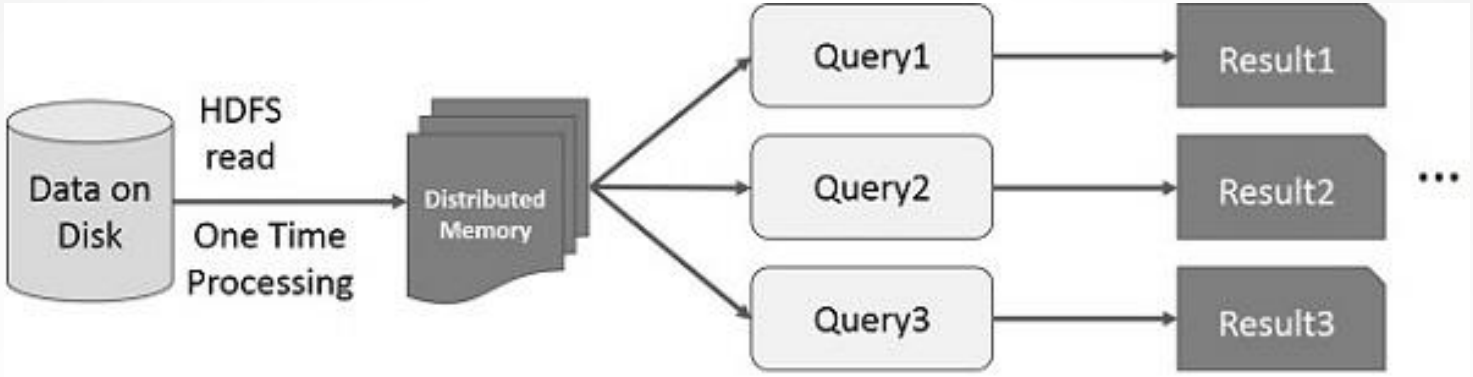
RDD, grafy obliczeniowe i odroczone wykonanie

Struktury danych - RDD

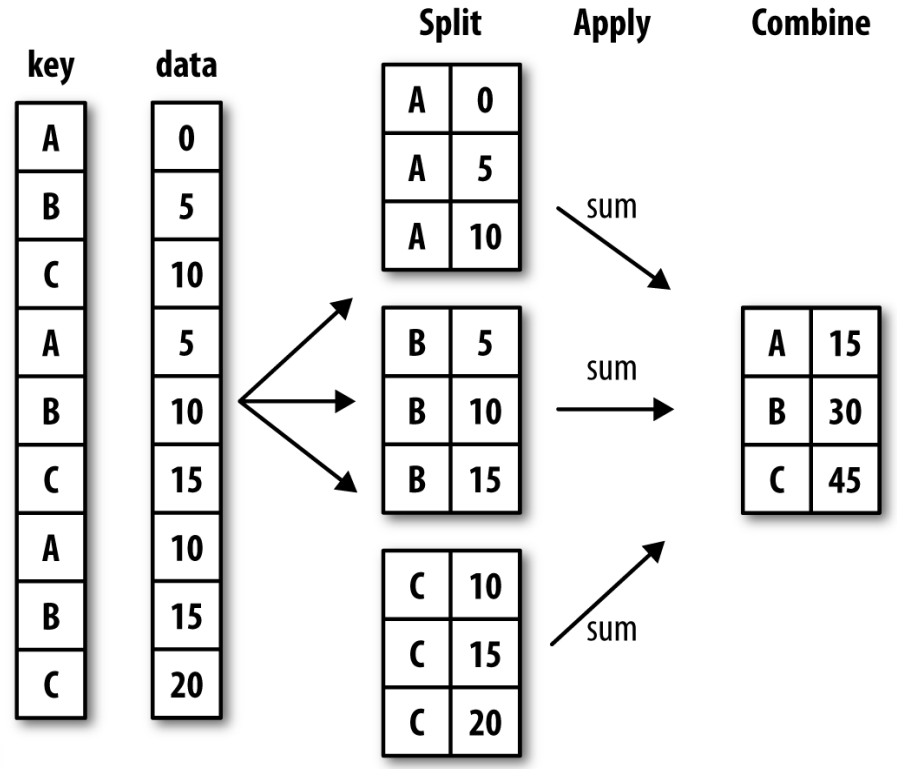
- Podstawową strukturą danych jest RDD – *Resilient Distributed Dataset,* wprowadzony w doktoracie Matei Zacharii
- Rozproszona struktura danych w postaci listy obiektów, indeksowanych kluczem
- Fragmenty listy mogą znajdować się w pamięci na różnych maszynach
- Obróbka fragmentów odbywa się w sposób rozproszony i równoległy
- Poszczególne kroki (*Stage)* są zachowywane i mogą być w razie potrzeby odtworzone – odporność na błędy (*Fault tolerance*)



Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, University of California, Berkeley,



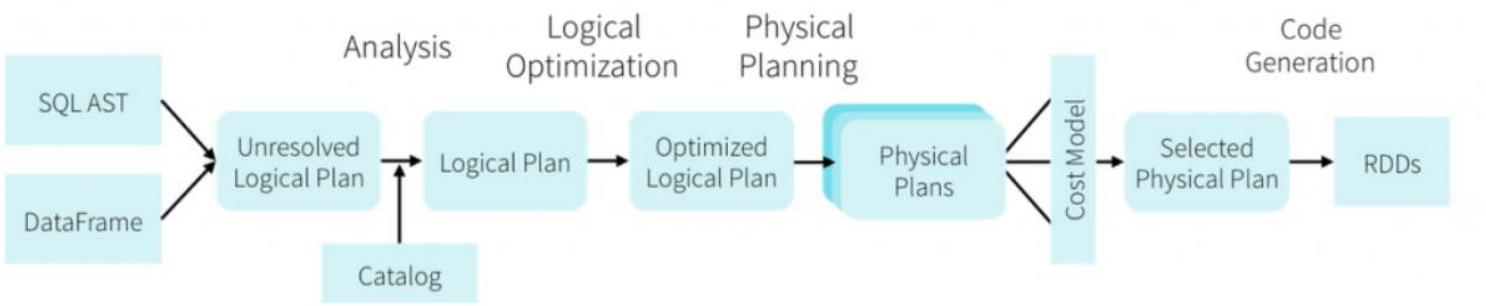
Karau, Holden. 2015. Learning Spark. Sebastopol: O'Reilly.



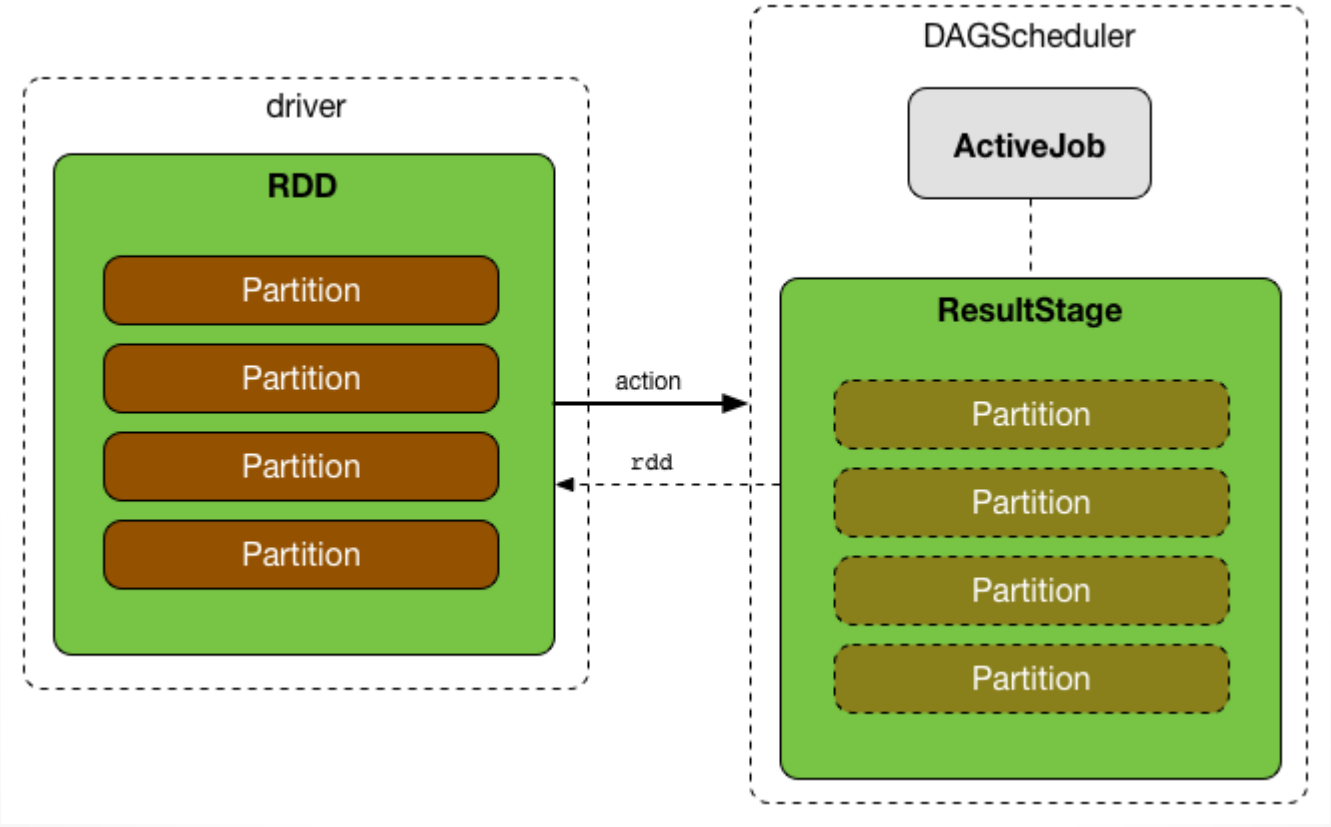
McKinney, Wes. 2018. Python for data analysis: data wrangling with pandas, NumPy, and IPython.

Operacje RDD

- Jednokrotne wczytanie danych do pamięci
- Tzw. *sharding,* czyli podział danych na partycje rozproszone – rozdzielone pomiędzy maszyny z egzekutorami
- Przetwarzanie kolejnych operacji w sposób rozproszony i równoległy
- Na końcu następuje scalenie wyników



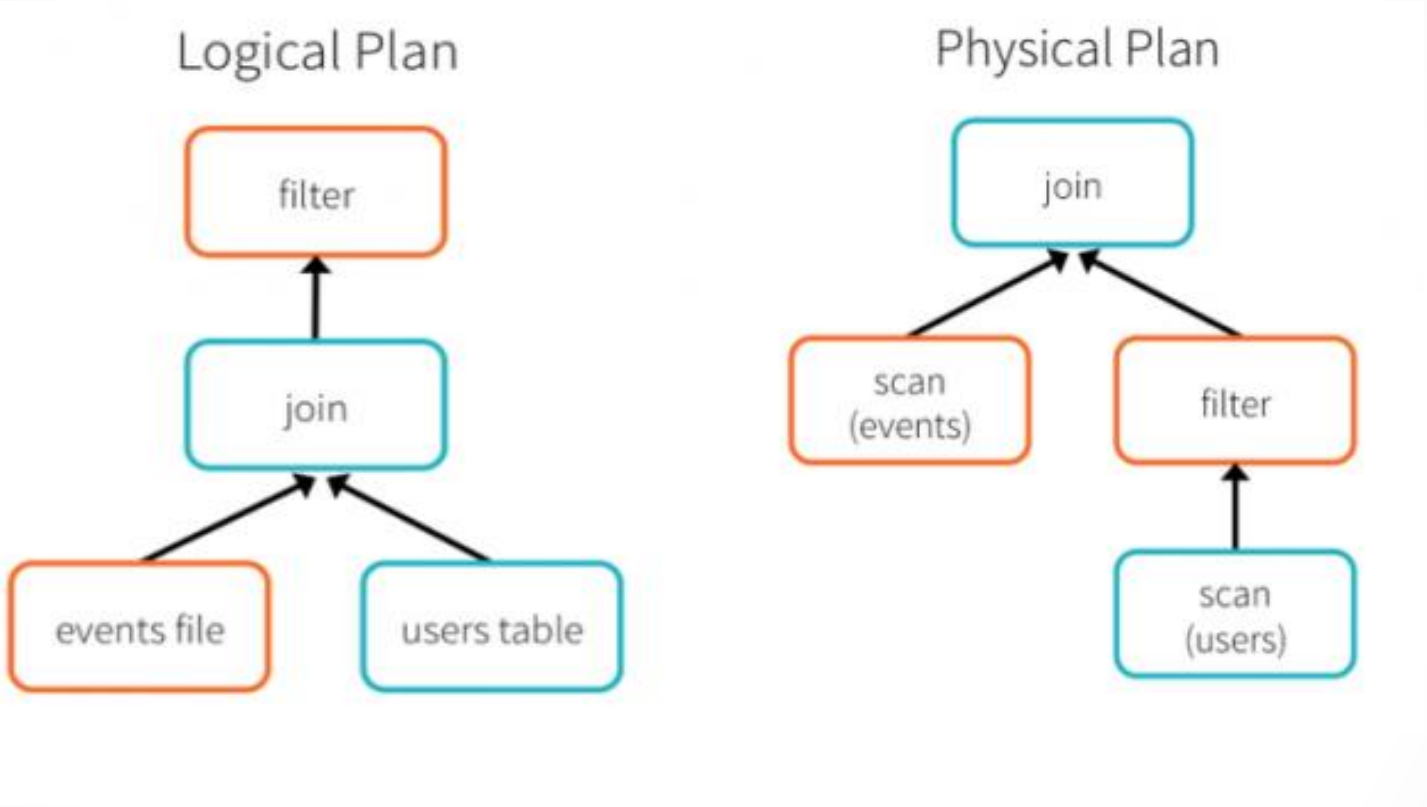
Apache Spark Analytics Made Simpe: http://go.databricks.com/apache-spark-analytics-made-simple-databricks



https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-dagscheduler.html

Grafy obliczeniowe

- Wykonywanie operacji nie odbywa się od razu
- Formułują one tzw. graf przekształceń (*transformation graph*) będący „przepisem”
- Podlega on optymalizacji przez silnik Sparka
- Na samym końcu następuje materializacja – tj. rzeczywiste wykonanie **zoptymalizowanego planu**



Grafy obliczeniowe

- Optymalizacja na poziomie **logicznym –** czyli wysokopoziomowych operacji zdefiniowanych przez użytkownika
- Optymalizacja na poziomie **fizycznym** – jak pobrać konkretne dane z konkretnych źródeł

Apache Spark Analytics Made Simpe: http://go.databricks.com/apache-spark-analytics-made-simple-databricks

Dziękuję za uwagę