Spark API

Dostępne języki

- Scala
- Java
- Python
- R

Spark napisany jest w Scali w związku z tym jest to język, który najwcześniej korzysta z nowo zaimplementowanych opcji.

Który język wybrać?

Wybór języka jest kwestią mocno subiektywną. Jednak najczęściej decyzja sprowadza się do pytania: Python czy Scala?

Java przegrywa z Pythonem i Scalą tym, że nie pozwala na interaktywną pracę z danymi (nie wspiera REPL), co jest bardzo ważne podczas pracy z danymi. Ponadto zwykle wymaga zdecydowanie więcej linijek kodu.

R pozostaje w tyle za pozostałymi językami – nowe funkcje pojawiają się tam z pewnym opóźnieniem. Nie jest to również język ogólnego zastosowania.

Python

- PYPL Index ->
- Aktywna społeczność
- Mnogość bibliotek
- Uniwersalność
- Czytelność
- Przystępność
- ...

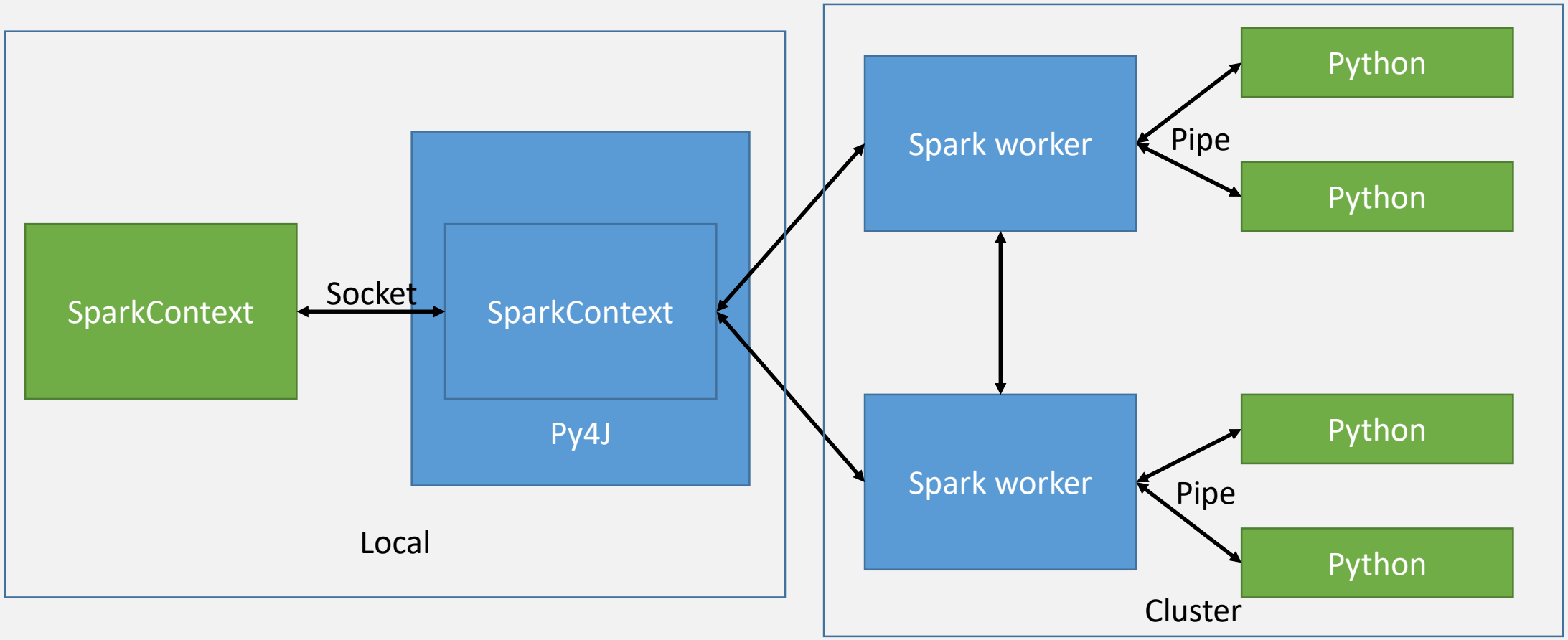
Worldwide, Nov 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	21.4 %	-1.9 %
2		Python	18.6 %	+5.2 %
3		PHP	8.2 %	-1.5 %
4	↑	Javascript	8.0 %	+0.5 %
5	↓	C#	7.6 %	-0.9 %
6	↑	C++	6.3 %	-0.7 %
7	↓	C	6.3 %	-0.9 %
8		Objective-C	3.9 %	-0.6 %
9		R	3.8 %	+0.6 %
10		Swift	3.1 %	+0.3 %
11		Matlab	2.2 %	-0.5 %
12		Ruby	1.8 %	-0.2 %
13	↑	VBA	1.5 %	+0.0 %
14	↑↑	TypeScript	1.4 %	+0.2 %
15		Scala	1.2 %	+0.1 %

Python

- Gdy uruchamiany jest interpreter Pythona uruchamiany jest też JVM z którym Python komunikuje się poprzez socket
- Pyspark wykorzystuje Py4J do obsługi tej komunikacji
- JVM działa jako właściwy Spark driver i ładuje JavaSparkContext, który komunikuje się z executorami na klastrze
- Działania na obiekcie SparkContext w Pythonie są tłumaczone na Javę i operują na JavaSparkContext
- Executory uruchamiają interpreter Pythona dla każdego rdzenia, z którymi komunikują się poprzez pipe kiedy muszą wykonać kod użytkownika

Python


SparkContext
Socket
SparkContext
Py4J
Local
Spark worker
Spark worker
Pipe
Pipe
Python
Python
Python
Python
Cluster

Spark API

API Sparka jest bardzo podobne w każdym języku, co ułatwia przechodzenie między językami.

Zilustrowane zostało to na kilku następnych slajdach przedstawiających proste zliczanie słów w tekście zaimplementowane w każdym z dostępnych języków.

WordCount – Java 7

```
JavaRDD<String> rawData = sc.textFile("hdfs://...");
JavaRDD<String> words = rawData.flatMap(new FlatMapFunction<String, String>() {
  public Iterable<String> call(String s) { return Arrays.asList(s.split(" ")); }
});
JavaPairRDD<String, Integer> pairs = words.mapToPair(
new PairFunction<String, String, Integer>() {
  public Tuple2<String, Integer> call(String s)
    { return new Tuple2<String, Integer>(s, 1); }
});
JavaPairRDD<String, Integer> counts = pairs.reduceByKey(new Function2<Integer, Integer>()
{
  public Integer call(Integer a, Integer b) { return a + b; }
});
counts.saveAsTextFile("hdfs://...");
```

WordCount – Java 8

```
JavaRDD<String> rawData = sc.textFile("hdfs://...");
JavaRDD<String> words =
  rawData.flatMap(line -> Arrays.asList(line.split(" ")));
JavaPairRDD<String, Integer> counts =
  words.mapToPair(w -> new Tuple2<String, Integer>(w, 1))
    .reduceByKey((x, y) -> x + y);
counts.saveAsTextFile("hdfs://...");
```

WordCount - Scala

```
val rawData = sc.textFile("hdfs://...")
val wordCounts = rawData.flatMap(line =>
                 line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_+_)
wordCounts.saveAsTextFile("hdfs://...")
```

WordCount - Python

```
rawData = sc.textFile("hdfs://...")
wordCounts = rawData.flatMap(lambda line:
                line.split(" "))
                .map(lambda word: (word, 1))
                .reduceByKey(lambda a, b: a + b)
wordCounts.saveAsTextFile("hdfs://...")
```

WordCount - R

```
rawData <- textFile(sc, "hdfs://... ")
words <- flatMap(rawData, function(line) {
                  strsplit(line, " ")[[1]] })
wordCount <- lapply(words, function(word) {
                    list(word, 1L) })
wordCounts <- reduceByKey(wordCount, "+", 2L)
saveAsTextFile(wordCounts, "hdfs://...")
```

Spark – pierwsze odpalenie

- Stwórz folder *sparkFun* (mkdir)
- Przejdź do folderu (cd *sparkFun*)
- Uruchom */opt/anaconda3/lib/python3.6/site-packages/pyspark/bin/spark-shell*
- Zaimplementuj WordCount w Scali
- Uruchom */opt/anaconda3/lib/python3.6/site-packages/pyspark/bin/pyspark*
- Zaimplementuj WordCount w Pythonie
- Plik znajuje się tu: */home/public/podyp_bd/spark/MobyDick.txt*