

AVATAR

FANTASTYKA



Jacek Kołodziej

Apache Spark - który
język wybrać: Scala vs.
PySpark

CAMERA

atm

ALLEGRO
TECH
MEETING 2021



BigData

BigData

- ...
- Apache Spark

BigData

- ...
- Apache Spark:
 - R
 - Java
 - Scala
 - Python

BigData

- ...
- Apache Spark:
 - R
 - Java
 - Scala
 - Python



kolodziejj.info
@unit03



Rozgrzewka

```
import org.apache.spark.sql.types.{DoubleType,  
IntegerType, StructField, StructType}  
import org.apache.spark.sql.{DataFrame, Row,  
SparkSession}  
import scala.collection.JavaConversions._
```

```
import org.apache.spark.sql.types.{DoubleType,
IntegerType, StructField, StructType}
import org.apache.spark.sql.{DataFrame, Row,
SparkSession}
import scala.collection.JavaConversions._

object WarmUp {
  def getSparkSession: SparkSession = {
    SparkSession
      .builder
      .master("local[*]")
      .appName("local-spark")
      .getOrCreate()
  }
}
```

```
import org.apache.spark.sql.types.{DoubleType,
IntegerType, StructField, StructType}
import org.apache.spark.sql.{DataFrame, Row,
SparkSession}
import scala.collection.JavaConversions._

object WarmUp {
  def getSparkSession: SparkSession = {
    SparkSession
      .builder
      .master("local[*]")
      .appName("local-spark")
      .getOrCreate()
  }
}
```

```
from pyspark.sql import (
    Row,
    SparkSession,
)

def get_spark_session() -> SparkSession:
    return (
        SparkSession
        .builder
        .master("local[*]")
        .appName("local-spark")
        .getOrCreate()
    )
```

...

```
object WarmUp {  
  ...  
  def createData(  
    sparkSession: SparkSession  
  ): DataFrame = {  
    val schema = StructType(  
      Array(  
        StructField("a", IntegerType),  
        StructField("b", DoubleType),  
      ),  
    )  
  
    sparkSession.createDataFrame(  
      Seq(  
        Row(2, 3.0),  
        Row(1, 2.5),  
        Row(2, 5.2),  
      ),  
      schema,  
    )  
  }  
}
```

...

```
object WarmUp {  
  ...  
  def createData(  
    sparkSession: SparkSession  
  ): DataFrame = {  
    val schema = StructType(  
      Array(  
        StructField("a", IntegerType),  
        StructField("b", DoubleType),  
      ),  
    )  
  
    sparkSession.createDataFrame(  
      Seq(  
        Row(2, 3.0),  
        Row(1, 2.5),  
        Row(2, 5.2),  
      ),  
      schema,  
    )  
  }  
}
```

...

```
object WarmUp {  
  ...  
  def createData(  
    sparkSession: SparkSession  
  ): DataFrame = {  
    val schema = StructType(  
      Array(  
        StructField("a", IntegerType),  
        StructField("b", DoubleType),  
      ),  
    )  
  
    sparkSession.createDataFrame(  
      Seq(  
        Row(2, 3.0),  
        Row(1, 2.5),  
        Row(2, 5.2),  
      ),  
      schema,  
    )  
  }  
}
```

...

```
object WarmUp {  
  ...  
  def createData(  
    sparkSession: SparkSession  
  ): DataFrame = {  
    val schema = StructType(  
      Array(  
        StructField("a", IntegerType),  
        StructField("b", DoubleType),  
      ),  
    )  
  
    sparkSession.createDataFrame(  
      Seq(  
        Row(2, 3.0),  
        Row(1, 2.5),  
        Row(2, 5.2),  
      ),  
      schema,  
    )  
  }  
}
```

...

```
object WarmUp {
  ...
  def createData(
    sparkSession: SparkSession
  ): DataFrame = {
    val schema = StructType(
      Array(
        StructField("a", IntegerType),
        StructField("b", DoubleType),
      ),
    )

    sparkSession.createDataFrame(
      Seq(
        Row(2, 3.0),
        Row(1, 2.5),
        Row(2, 5.2),
      ),
      schema,
    )
  }
}
```

...

```
def create_data(
  spark: SparkSession,
) -> DataFrame:
  schema = StructType(
    [
      StructField("a", IntegerType()),
      StructField("b", DoubleType()),
    ],
  )

  return spark.createDataFrame(
    [
      Row(a=2, b=3.),
      Row(a=1, b=2.5),
      Row(a=2, b=5.2),
    ],
    schema,
  )
```

```
...  
  
object WarmUp {  
  ...  
  def main(args: Array[String]): Unit = {  
    val spark: SparkSession = getSparkSession  
  
    val df: DataFrame = createData(spark)  
  
    df.printSchema()  
    df.show()  
  
    spark.stop()  
  }  
}
```

...

```
object WarmUp {  
  ...  
  def main(args: Array[String]): Unit = {  
    val spark: SparkSession = getSparkSession  
  
    val df: DataFrame = createData(spark)  
  
    df.printSchema()  
    df.show()  
  
    spark.stop()  
  }  
}
```

...

```
if __name__ == "__main__":  
    spark = get_spark_session()  
  
    df = create_data(spark)  
  
    df.printSchema()  
    df.show()  
  
    spark.stop()
```

```
echo 'name := "Foo"

version := "1.0"

scalaVersion := "2.12.10"

libraryDependencies += "org.apache.spark" %%
"spark-sql" % "3.1.2"
libraryDependencies += "org.scalatest" %%
"scalatest" % "3.2.9" % "test"
' > build.sbt

sbt package && \
  spark-submit \
    --class "WarmUp" \
    target/scala-2.12/foo_2.12-1.0.jar
```

```
echo 'name := "Foo"
```

```
version := "1.0"
```

```
scalaVersion := "2.12.10"
```

```
libraryDependencies += "org.apache.spark" %%
```

```
"spark-sql" % "3.1.2"
```

```
libraryDependencies += "org.scalatest" %%
```

```
"scalatest" % "3.2.9" % "test"
```

```
' > build.sbt
```

```
sbt package && \
```

```
  spark-submit \
```

```
    --class "WarmUp" \
```

```
    target/scala-2.12/foo_2.12-1.0.jar
```

```
pip install pyspark
```

```
python src/warm_up.py
```

```
[...coś o kompilacji...]  
[...jakieś warningi...]  
[...jakieś info...]  
root  
|-- a: integer (nullable = true)  
|-- b: double (nullable = true)
```

```
[...jakieś info...]
```

```
+---+---+  
|  a|  b|  
+---+---+  
|  2|3.0|  
|  1|2.5|  
|  1|5.2|  
+---+---+
```

```
[...coś o kompilacji...]  
[...jakieś warningi...]  
[...jakieś info...]  
root  
|-- a: integer (nullable = true)  
|-- b: double (nullable = true)
```

```
[...jakieś info...]  
+---+---+  
|  a|  b|  
+---+---+  
|  2|3.0|  
|  1|2.5|  
|  1|5.2|  
+---+---+
```

```
[...jakieś warningi...]  
root  
|-- a: integer (nullable = true)  
|-- b: double (nullable = true)
```

```
+---+---+  
|  a|  b|  
+---+---+  
|  2|3.0|  
|  1|2.5|  
|  2|5.2|  
+---+---+
```

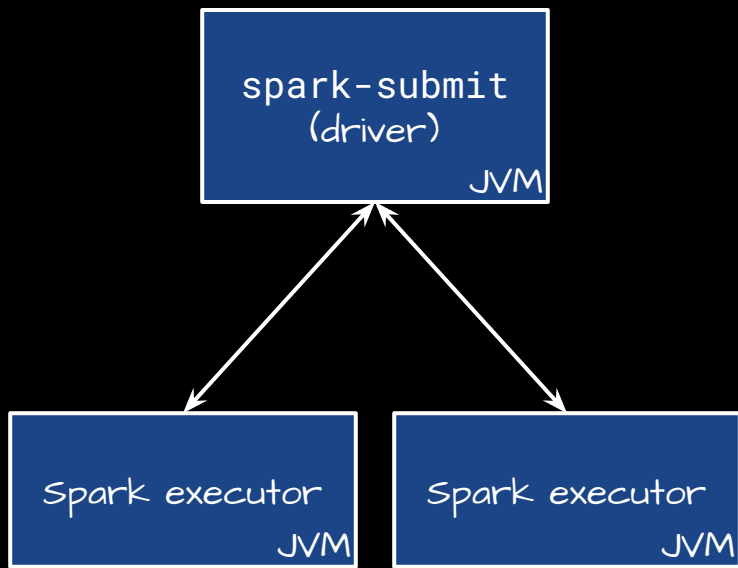
```
[...coś o kompilacji...]  
[...jakieś warningi...]  
[...jakieś info...]  
root  
|-- a: integer (nullable = true)  
|-- b: double (nullable = true)
```

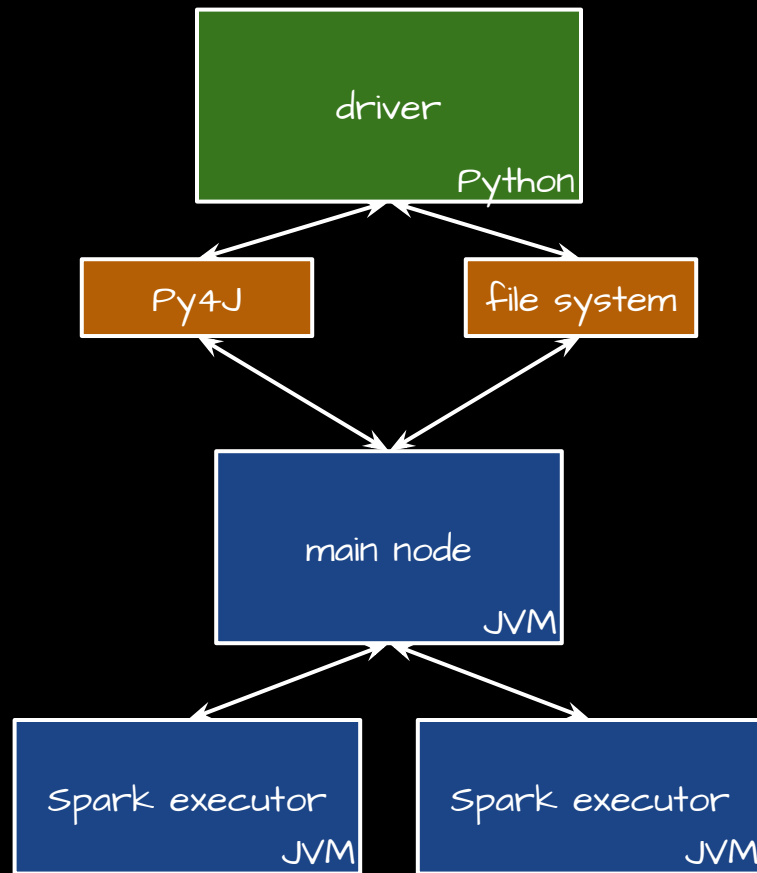
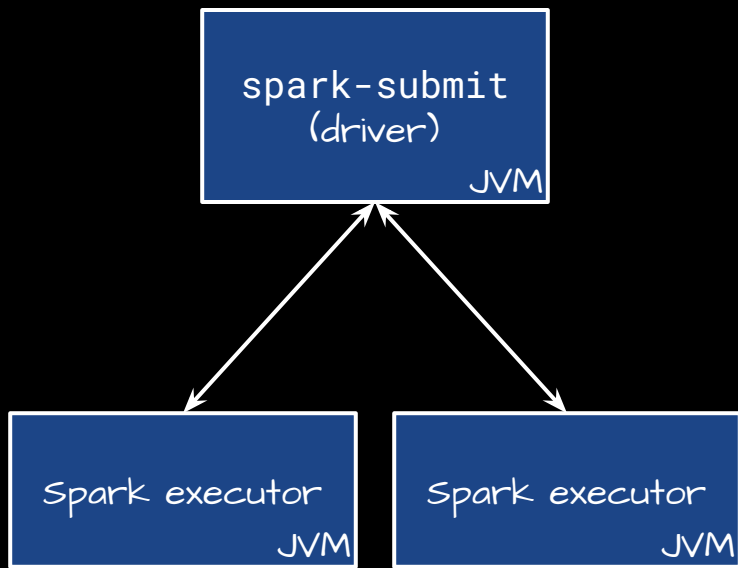
```
[...jakieś info...]
```

```
+---+---+  
|  a|  b|  
+---+---+  
|  2|3.0|  
|  1|2.5|  
|  1|5.2|  
+---+---+
```

```
[...jakieś warningi...]  
root  
|-- a: integer (nullable = true)  
|-- b: double (nullable = true)
```

```
+---+---+  
|  a|  b|  
+---+---+  
|  2|3.0|  
|  1|2.5|  
|  2|5.2|  
+---+---+
```





Development

Testy

Testy

```
import org.apache.spark.sql.{Row, SparkSession}
import org.scalatest.flatspec.AnyFlatSpec
import org.scalatest.matchers.should

class WarmUpSpec extends AnyFlatSpec with
  should.Matchers {
    private def getSparkSession: SparkSession = {
      SparkSession
        .builder
        .master("local[*]")
        .appName("local-spark")
        .getOrCreate()
    }
}
```

Testy

```
import org.apache.spark.sql.{Row, SparkSession}
import org.scalatest.flatspec.AnyFlatSpec
import org.scalatest.matchers.should
```

```
class WarmUpSpec extends AnyFlatSpec with
should.Matchers {
  private def getSparkSession: SparkSession = {
    SparkSession
      .builder
      .master("local[*]")
      .appName("local-spark")
      .getOrCreate()
  }
}
```

```
from pyspark.sql import (
    DataFrame, Row, SparkSession,
)
import pytest
```

```
from warm_up import (
    group_and_count,
    create_data,
)
```

```
@pytest.fixture
def spark_session() -> SparkSession:
    return (
        SparkSession
        .builder
        .master("local[*]")
        .appName("local-spark")
        .getOrCreate()
    )
```

Testy

```
class WarmUpSpec extends ... {  
  ...  
  
  "WarmUp" should "group and count" in {  
    val spark: SparkSession = getSparkSession  
    val df: DataFrame = WarmUp.createData(spark)  
  
    WarmUp  
      .groupAndCount(df)  
      .collect() should contain  
theSameElementsAs Seq(  
    Row(2, 2), Row(1, 1),  
  )  
  }  
}
```

Testy

```
class WarmUpSpec extends ... {  
  ...  
  
  "WarmUp" should "group and count" in {  
    val spark: SparkSession = getSparkSession  
    val df: DataFrame = WarmUp.createData(spark)  
  
    WarmUp  
      .groupAndCount(df)  
      .collect() should contain  
theSameElementsAs Seq(  
    Row(2, 2), Row(1, 1),  
  )  
  }  
}
```

```
@pytest.fixture  
def sample_data(spark_session) -> DataFrame:  
    return read_data(spark_session)  
  
def test_group_and_count(  
    sample_data: DataFrame  
) -> None:  
    assert (  
        set(  
            group_and_count(sample_data)  
              .collect()  
        )  
        == {  
            Row(a=1, count=1),  
            Row(a=2, count=2),  
        }  
    )
```

Typowanie

- silne
- statyczne

Typowanie

- silne
- statyczne
- => dużo błędów wykrywanych przy kompilacji

Typowanie

- silne
- statyczne
- => dużo błędów wykrywanych przy kompilacji

- silne

Typowanie

- silne
 - statyczne
 - => dużo błędów wykrywanych przy kompilacji
- silne
 - dynamiczne

Typowanie

- silne
- statyczne
- => dużo błędów wykrywanych przy kompilacji

- silne
- dynamiczne
- => potrzeba więcej testów i statycznej analizy

Typowanie

- silne
- statyczne
- => dużo błędów wykrywanych przy kompilacji

- silne
- dynamiczne
- => potrzeba więcej testów i statycznej analizy
- opcjonalne type hints
 - osobiście mocno polecam!

API Sparkowe

- RDD (Resilient Distributed Datasets)
- DataFrame'y

API Sparkowe

- RDD (Resilient Distributed Datasets)
- DataFrame'y
- Datasetsy (statycznie typowane)

API Sparkowe

- RDD (Resilient Distributed Datasets)
- DataFrame'y
- Datasetsy (statycznie typowane)
 - tylko w Scali :(

API Sparkowe

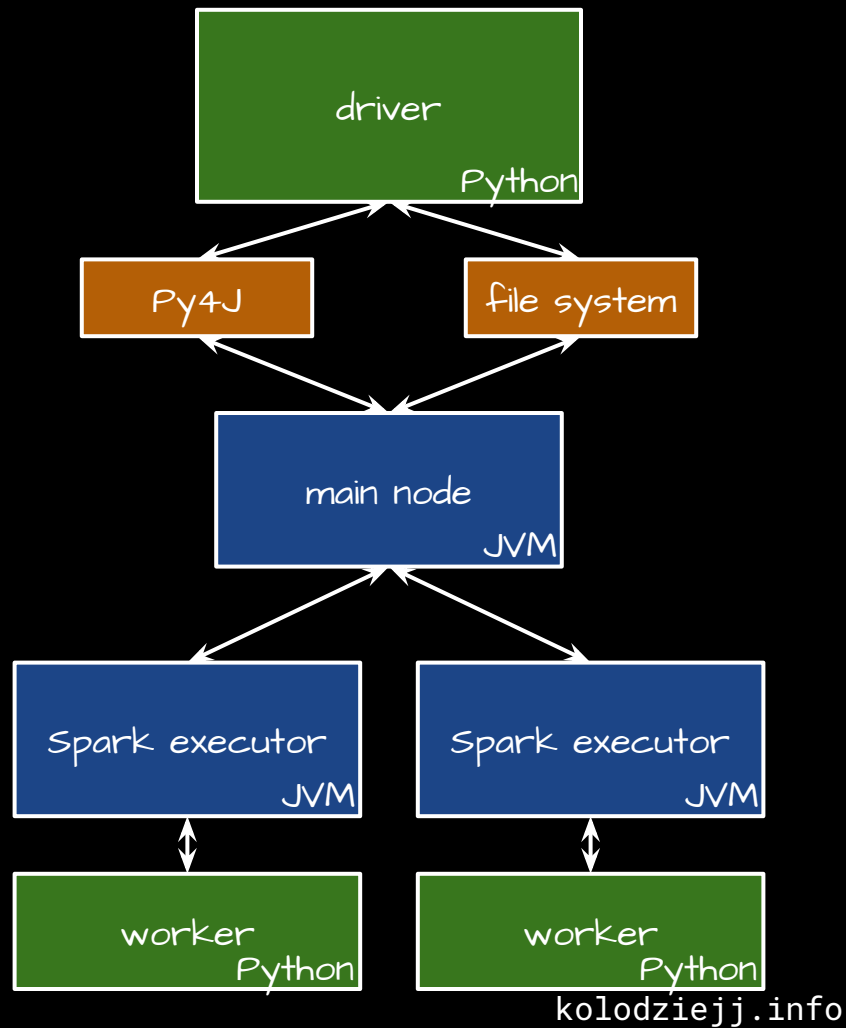
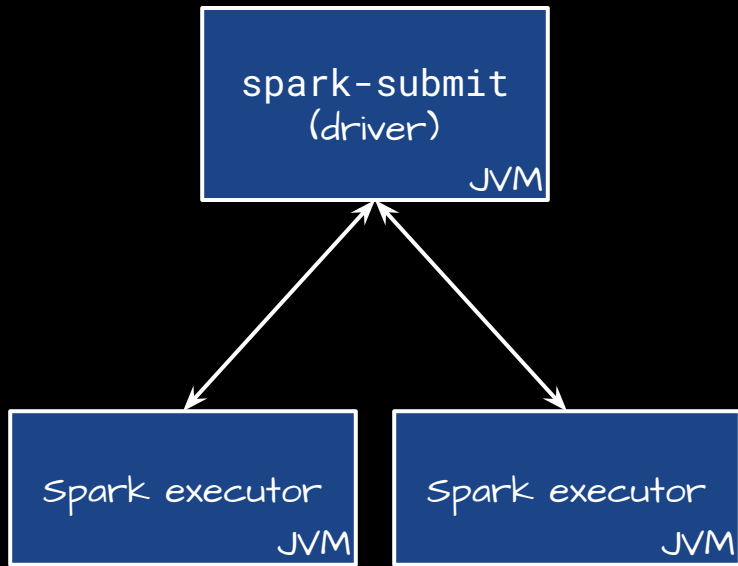
- RDD (Resilient Distributed Datasets)
- DataFrame'y
- Datasetsy (statycznie typowane)
 - tylko w Scali :(
- Spark SQL

API Sparkowe

- RDD (Resilient Distributed Datasets)
- DataFrame'y
- Datasets (statycznie typowane)
 - tylko w Scali :(
- Spark SQL
- UDFy
 - Scala: też w JVM

API Sparkowe

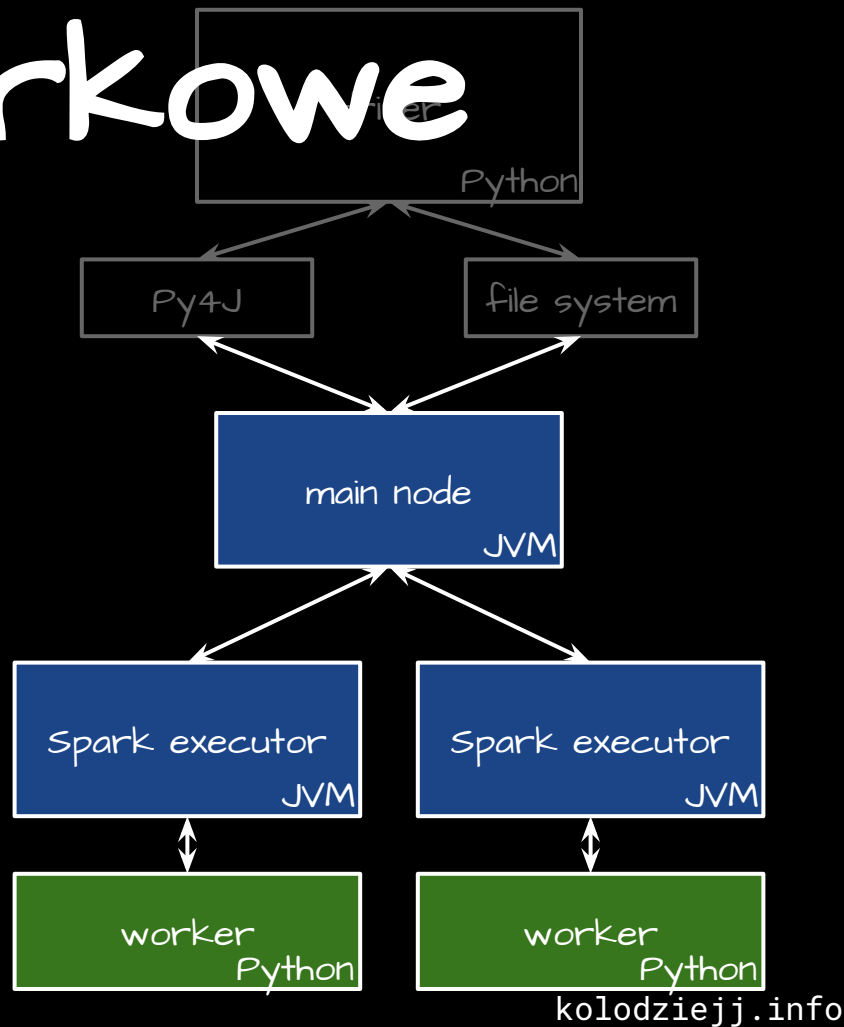
- RDD (Resilient Distributed Datasets)
- DataFrame'y
- Datasets (statycznie typowane)
 - tylko w Scali :(
- Spark SQL
- UDFy
 - Scala: też w JVM
 - PySpark...



API Sparkowe

PySpark:

- DataFrame'y
- Spark SQL
- UDFy i operacje na RDD z dodatkowym narzutem na (de)serializację :(
- bez Datasetów :(



Ekosystem

Ekosystem

- Spark:
 - Structured Streaming
 - MLlib
- Spark:
 - Structured Streaming
 - MLlib

Ekosystem

- Spark:

- Structured Streaming
- MLlib
- GraphX

- Spark:

- Structured Streaming
- MLlib
- ~~○ GraphX~~

Ekosystem

- Spark:
 - Structured Streaming
 - MLlib
 - GraphX
 - Scala first
- Spark:
 - Structured Streaming
 - MLlib
 - ~~GraphX~~

Ekosystem

- Spark:
 - Structured Streaming
 - MLlib
 - GraphX
 - Scala first
- Spark:
 - Structured Streaming
 - MLlib
 - ~~GraphX~~
- NumPy, Pandas, Koalas...

Ekosystem

- Spark:
 - Structured Streaming
 - MLlib
 - GraphX
 - Scala first
- Spark:
 - Structured Streaming
 - MLlib
 - ~~GraphX~~
- NumPy, Pandas, Koalas...
- TensorFlow, ...
- ...

Ekosystem

- Spark:
 - Structured Streaming
 - MLlib
 - GraphX
 - Scala first
- Breeze, Spire, ...
- DeepLearning4J, BigDL, H2O, ...
- ...
- Spark:
 - Structured Streaming
 - MLlib
 - ~~GraphX~~
- NumPy, Pandas, Koalas...
- TensorFlow, ...
- ...

Ekosystem

- Co daje większą wartość w danym przypadku?

Ekosystem

- Co daje większą wartość w danym przypadku?
- Co zna/chce poznać nasz zespół i otoczenie?

Wdrażanie

Wdrażanie

- zależności w JARze
- zależności w
 - ZIPie
 - spakowanym virtual envie
 - i/lub zainstalowane na workerze

Wdrażanie

- zależności w JARze

- zależności w

- ZIPie
- spakowanym virtual envie
- i/lub zainstalowane na workerze

```
spark-submit \  
  --class MyClass \  
  sciezka/do.jar \  
  jakies \  
  argumenty
```

```
spark-submit \  
  --py-files ... \  
  --archives ... \  
  sciezka/do.py \  
  jakies \  
  argumenty
```

utrzymanie

Utrzymanie

- Kompetencje w zespole

Utrzymanie

- kompetencje w zespole
- współpraca z innymi

Utrzymanie

- kompetencje w zespole
- współpraca z innymi
- typowanie

Utrzymanie

- kompetencje w zespole
- współpraca z innymi
- typowanie
- narzędzia:
 - statyczna analiza
 - profilowanie

Popularność

Popularność

- JetBrains 2020 Developer Survey
 - Scala Spark:
 - PySpark:
- StackOverflow 2021 Survey
 - Scala Spark:
 - PySpark:

Popularność

- JetBrains 2020 Developer Survey
 - Scala Spark: 268
 - PySpark: 702
- StackOverflow 2021 Survey
 - Scala Spark: 840
 - PySpark: 2183

Popularność

- JetBrains 2020 Developer Survey
 - Scala Spark: 268
 - PySpark: 702
- StackOverflow 2021 Survey
 - Scala Spark: 840
 - PySpark: 2183
- Allegro

Wydajność

Wydajność

- *select, zapis*
 - Scala Spark:
 - PySpark:

Wydajność

- *select, zapis*
 - Scala Spark: ~1,7 min
 - PySpark: ~1,7 min

Wydajność

- `select`, zapis
 - Scala Spark: ~1,7 min
 - PySpark: ~1,7 min
- `select`, dodanie kolumny przez UDF, zapis
 - Scala Spark: ~2,5 min

Wydajność

- `select, zapis`
 - Scala Spark: ~1,7 min
 - PySpark: ~1,7 min
- `select, dodanie kolumny przez UDF, zapis`
 - Scala Spark: ~2,5 min
 - PySpark: ~4,1 min

Wydajność

- `select, zapis`
 - Scala Spark: ~1,7 min
 - PySpark: ~1,7 min
- `select, dodanie kolumny przez UDF, zapis`
 - Scala Spark: ~2,5 min
 - PySpark: ~4,1 min
- uwaga na takie benchmarki :)

Jak wybrać?

Jak wybrać?

- wymagania funkcjonalne
 - raczej narzędzia niż sam język

Jak wybrać?

- wymagania funkcjonalne
 - raczej narzędzia niż sam język
- wymagania wydajnościowe
 - ścisłe, czy tylko ambicjonalne?

Jak wybrać?

- wymagania funkcjonalne
 - raczej narzędzia niż sam język
- wymagania wydajnościowe
 - ścisłe, czy tylko ambicjonalne?
- kompetencje (i ambicje!) maintainerów
 - aktualnych i przyszłych

Jak wybrać?

- wymagania funkcjonalne
 - raczej narzędzia niż sam język
- wymagania wydajnościowe
 - ścisłe, czy tylko ambicjonalne?
- kompetencje (i ambicje!) maintainerów
 - aktualnych i przyszłych
- testowanie, typowanie, czas kompilacji etc.

Dzięki! :)

Pytania?

kolodziej.j.info/talks/pyspark-scala