Why Did We Create Kotlin API for Apache Spark

Pasha Finkelshteyn, JetBrains

Pasha Finkelshteyn

Developer 🥑 for Big Data @ JetBrains

@asm0di0

How did I fall in love with Big Data?

- 7 years ago Hadoop looked like a magic to Java enterprise developer
- Started to look for data-related projects
- Started to read about DBs
- Moved from Team Lead to Data Engineer

First thing Data Engineer learns?

What did I have

- Java
- Kotlin
- Scala
- Groovy
- bash, XML, YAML 😄

Sparse experience with other languages

What did I have

- Core Java
- GC
- Lots of debug experience
- Distributed systems
- Architecture

Supported languages

The most popular language in data engineering

```python
from pyspark.sql import DataFrame, functions as F
from pyspark.sql.types import *

data_type = StructType([
    StructField("pk", LongType(), False),
    StructField("aa", StringType(), False)
  ])

transformed_data = (
  sample_df
  .withColumn("json_object", F.get_json_object(F.col("json_value"), "$.data"))
  .withColumn("parsed_struct", F.from_json(F.col("json_object"), data_type))
  .selectExpr("parsed_struct.*")
)
```

```python
from pyspark.sql import DataFrame, functions as F
from pyspark.sql.types import *

data_type = StructType([
    StructField("pk", LongType(), False),
    StructField("aa", StringType(), False)
  ])

transformed_data = (
  sample_df
  .withColumn("json_object", F.get_json_object(F.col("json_value"), "$.data"))
  .withColumn("parsed_struct", F.from_json(F.col("json_object"), data_type))
  .selectExpr("parsed_struct.*")
)
```

```python
from pyspark.sql import DataFrame, functions as F
from pyspark.sql.types import *

data_type = StructType([
    StructField("pk", LongType(), False),
    StructField("aa", StringType(), False)
  ])

transformed_data = (
  sample_df
  .withColumn("json_object", F.get_json_object(F.col("json_value"), "$.data"))
  .withColumn("parsed_struct", F.from_json(F.col("json_object"), data_type))
  .selectExpr("parsed_struct.*")
)
```

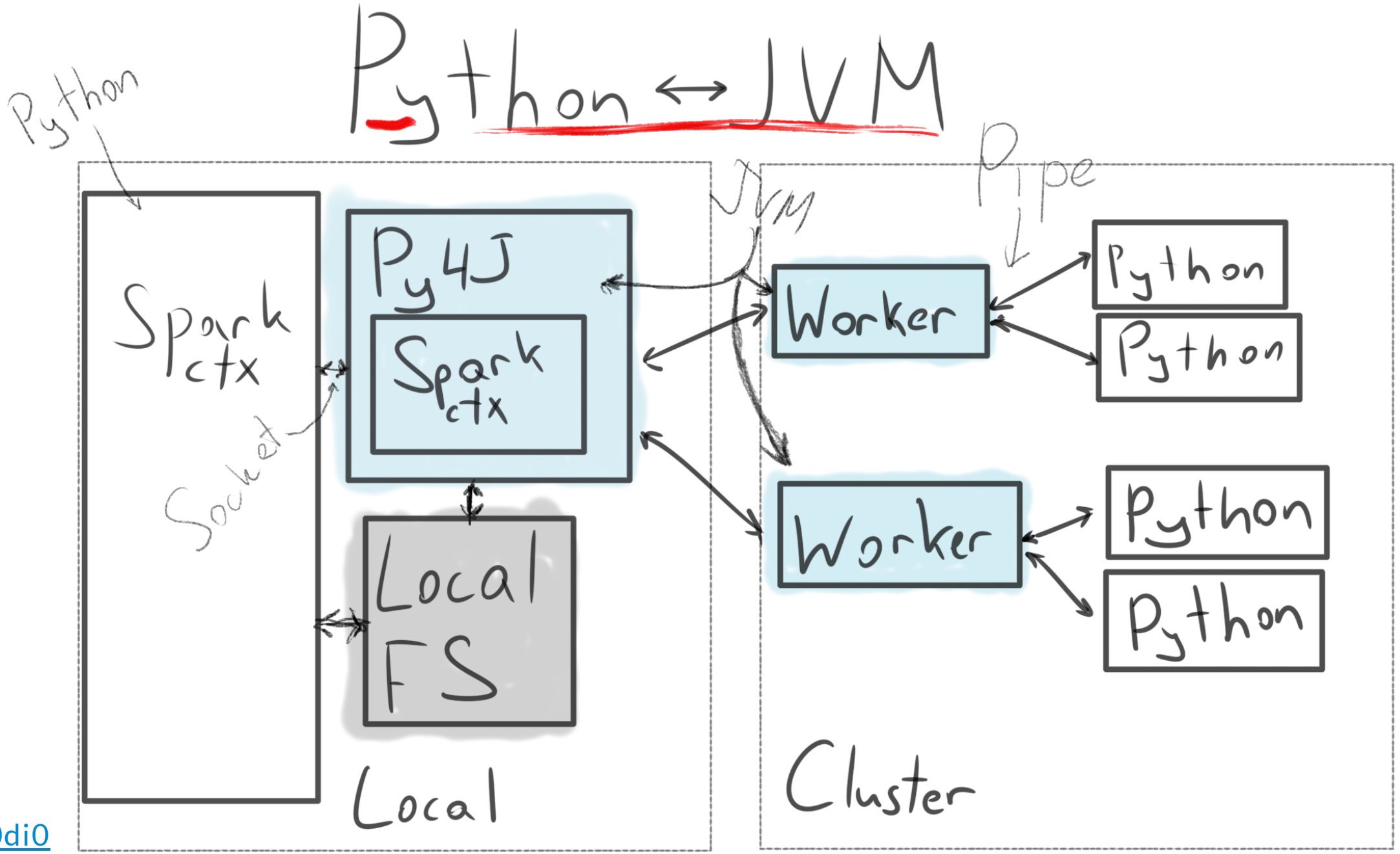
```python
from pyspark.sql import DataFrame, functions as F
from pyspark.sql.types import *

data_type = StructType([
    StructField("pk", LongType(), False),
    StructField("aa", StringType(), False)
  ])

transformed_data = (
  sample_df
  .withColumn("json_object", F.get_json_object(F.col("json_value"), "$.data"))
  .withColumn("parsed_struct", F.from_json(F.col("json_object"), data_type))
  .selectExpr("parsed_struct.*")
)
```

Python

- `F` is made to distinct Spark built-ins from self-made.
  Who have ever created their own function over column?
- API is untyped
- Everything is string-based
- `UDF`s are SLOW
- Custom type support is complex


Python ↔ JVM
Python
Spark ctx
Socket
Py4J
Spark ctx
Local FS
Local
JVM
Pipe
Worker
Worker
Python
Python
Python
Python
Cluster

Python

```
checkResult = VerificationSuite(spark) \
    .onData(df) \
    .addCheck(
        check.hasSize(lambda x: x ⩾ 3) \
        .hasMin("b", lambda x: x == 0) \
        .isComplete("c")  \
        .isUnique("a")  \
        .isContainedIn("a", ["foo", "bar", "baz"]) \
        .isNonNegative("b")) \
    .run()
```

Scala

The best* official API

- Typed and untyped APIs
- Awesome smart encoders
- Spark is written in Scala:
  - Best interop possible
- Huge ecosystem

* by my own rating among official APIs

The hard parts

```
implicit val z1 = 2
def addTo(n: Int) = {
    def add(x: Int)(y: Int)(implicit z: Int) = x + y + z
    add(n) _
}
val addTo1 = addTo(1)
addTo1(2)
```

The hard parts

```
trait HashSet[+T] {
  def add[U >: T](item: U)
}
```

The hard parts

```
trait Zulimba[+T] {
  def gogoThere[U >: T](next: U)
}
```

The hard parts

- Hard to learn for people without JVM knowledge
- Hard to read code somebody else wrote
- Easy to abuse language features
  - operator overloading
  - implicit
  - traits

Scala's type system is awesome and powerful!

Scala's type system is awesome and powerful!

Yes, but even Spark doesn't utilize it's full power

Scala's type system is awesome and powerful!

Yes, but even Spark doesn't utilize it's full power

For the greater good