Python Streaming API

Zohar Mizrahi
Senior Software Architect
ParallelM

Flink Forward 2017,
Berlin, Germany

What Will Be Covered

- ParallelM
- Python in Machines Learning
- Python Batch API
- Python Streaming API
- Live Demo

ParallelM

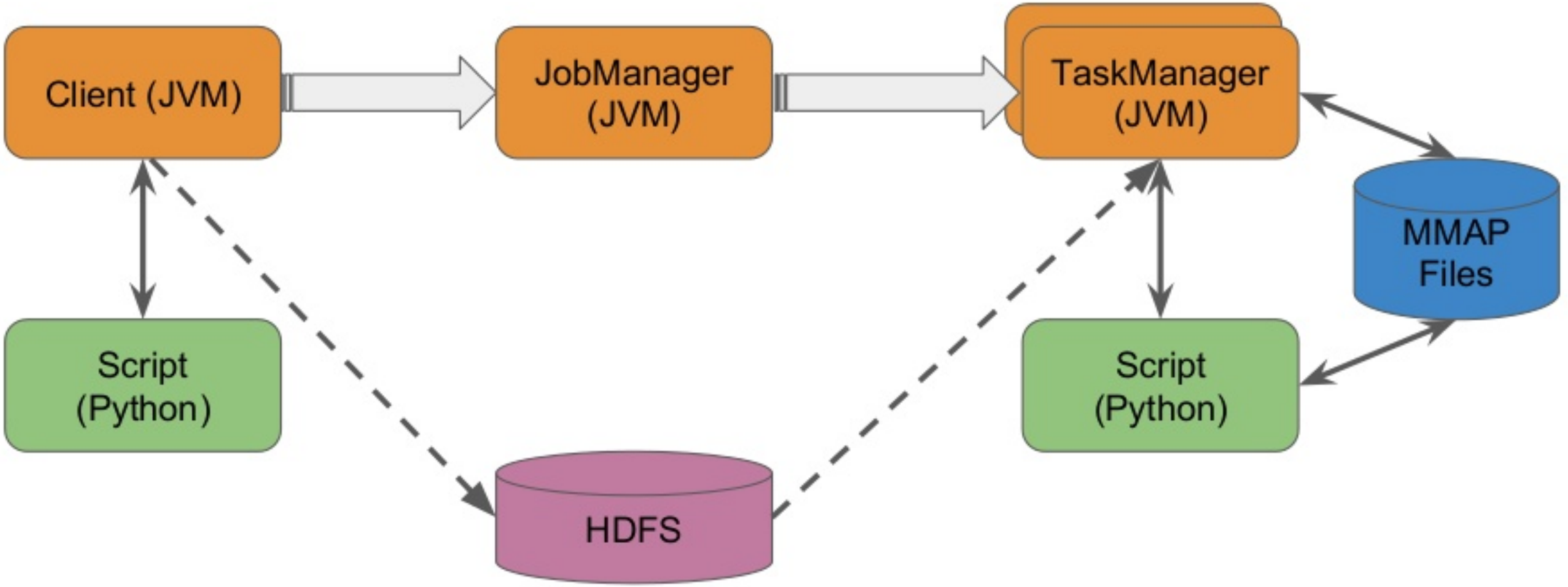
ParallelM accelerates time to value of AI initiatives
by helping ML Ops and Data Science teams
deploy and manage Machine Learning (ML) in Production

We have put much effort in Flink, because of its exceptional design and ability to handle high speed real time and true stream processing

Python In Machine Learning

- Popular in data analysis (NumPy, SciPy, Matplotlib, Panda, etc.)
- Very easy to learn
- Very easy to read
- Does not require compilation
- Awesome online community

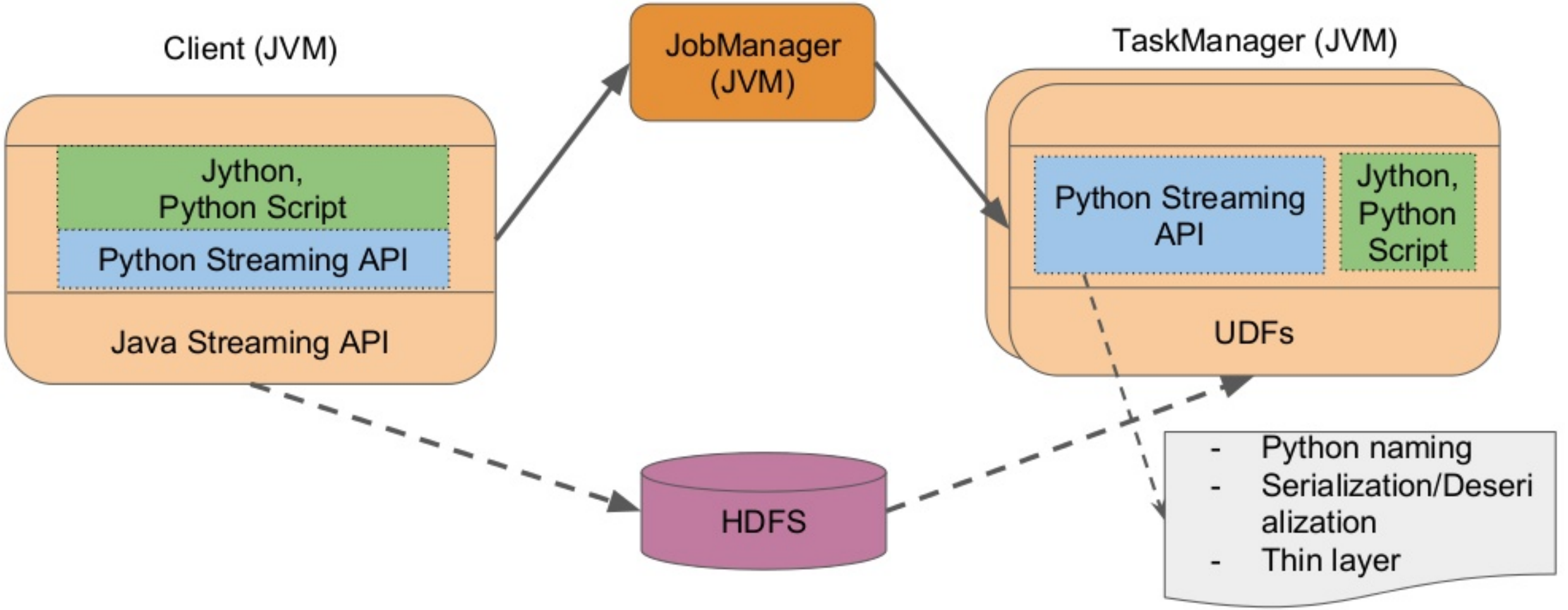
Python Batch Processing Overview



… but **Flink** is a **Stream Processing** framework …

What about **Python** for Streaming Data?

Proposed Python Streaming API Architecture



Jython

- Python engine in java (http://www.jython.org/)
- Possible to use CPython extensions like NumPy or SciPy with **JyNI** (http://jyni.org/)
- Glitches
  - The latest supported Python version is 2.7
  - No official statement for coming support in Python 3.x

What Jython Challenges I had to solve?

- Java class serial ID mismatch
  - Execution of the same script multiple times
- Different namespaces for python classes with same name but different code
  - Execution of different scripts having the same class name
- Python paths and imports issues
  - A python script may import additional files and folder

Performance Considerations

- Initialisation of the **Jython** framework impose a fixed overhead of 2 ~ 5 seconds:
  - Client - whenever submitted
  - TaskManager - only once, on the first submitted job
- Java/Scala vs. Python
  - No high-scale tests were conducted yet
  - There's a room for optimizations

Python Script - main

```
def main():
    env = PythonStreamExecutionEnvironment.get_execution_environment()

    env.read_text_file("/tmp/book.txt") \
        .flat_map(Tokenizer()) \
        .key_by(Selector()) \
        .time_window(milliseconds(50)) \
        .reduce(Sum()) \
        .print()

    env.execute(True)
```

Python Script - UDF

```
class Tokenizer(FlatMapFunction):
    def flatMap(self, value, collector):
        for word in value.lower().split():
            collector.collect((1, word))

class Selector(KeySelector):
    def getKey(self, input):
        return input[1]

class Sum(ReduceFunction):
    def reduce(self, input1, input2):
        count1, word1 = input1
        count2, word2 = input2
        return (count1 + count2, word1)
```

Status / API Coverage

- Pending pull request (**#3838**)
- New project under:
  `flink-libraries/flink-streaming-python`
- Partial coverage of the whole streaming API (Beta)

Tests / Examples

- Internal tests are under:

flink-libraries/flink-streaming-python/src/test/python/org/apache/flink/streaming/python/api

- One complete example:

flink-examples/flink-examples-streaming/src/main/python/fibonacci.py

How to execute

```
> ./bin/pyflink-stream.sh /tmp/fibonacci.py - --local
```

Notes:

- New command line tool: **`pyflink-stream.sh`**
- Command line arguments: **`after the dash(`-`)`**
- For local execution: `env.execute(True)`
- Cluster mode requires **HDFS**

Demo Time

- Fibonacci python example
- Functionality
  - Calculates fibonacci series up to an upper bound
- Input
  - “<x>, <y>” - stream of pairs of numbers
- Output
  - ((<x>, <y>), <#iters>) - the original pair along with the iterations number