Performance Comparison Between Scala User-Defined Functions and Python User-Defined Functions in Spark

Titus An

May 12, 2018

Introduction

This document describes the result obtained from comparing the performance of user-defined functions (UDF) under Scala and Python in Spark. It can be shown that invoking Spark UDF from Python has a performance boost around factor of two comparing to invoking Python UDF. In conclusion, it is desirable for frameworks to have their UDFs implemented in Scala, and expose those functions to Python scripts to be used together with 'select' method of Data Frames. This provides the fastest execution.

Methodology

Benchmarks are performed on web archive data provided by the University of Toronto Libraries. It is 95.9MB in size. A user defined function is applied to uniform resource locator (URL) field of data. The function extracts domain name (without dot www part) from the URL as its output. A series of transformations is then applied to domain names. Specifically, a list of most frequent domain appeared is collected, along with their corresponding number of occurrences.

Two identical version of the function that extracts domain name from URL is implemented in both Python and Scala.

```python
def extractDomain(url):
    url = url.replace('http://', '').replace('https://', '')
    if '/' in url:
        return url.split('/')[0].replace('www.', '')
    else:
        return url.replace('www.', '')
```

Python version

```scala
def extractDomain(url: String) = {
  val result = url.replace("https://", "").replace("http://", "")
  if (result contains '/') {
    result.split('/')(0).replace("www.", "")
  } else {
    result.replace("www.", "")
  }
}
```

Scala version

Five different scenarios are used to thoroughly test the performance of two UDF implementations. Each test runs on its own, so that caches from previous runs cannot be taken advantage of. Each test is repeated ten times, and the running time from the start of 'select' keyword to the end of 'head' is collected. The time is in number of milliseconds.

Tests

Scala program calls Scala UDF with function call (SSF)

```
var result = df
             .select(ExtractDomainUDF($"Url").as("Domain"))
             .groupBy("Domain")
             .count()
             .orderBy(desc("count"))
             .head(3)
```

Expression used

```
d1=c(5285,5034,5250,5484,4980,5534,5695,5244,5117,5152)
m1=mean(d1)
s1=sd(d1)
```

The mean process time is 5277.5 ms.

Scala program calls Scala UDF with SQL expression (SSS)

```
var result = df
             .selectExpr("extractDomainReg(Url) as Domain")
             .groupBy("Domain")
             .count()
             .orderBy(desc("count"))
             .head(3)
```

Expression used

```
d2=c(5771,5482,5607,5416,5382,5927,5390,5349,5310,5616)
m2=mean(d2)
s2=sd(d2)
```

The mean process time is 5525 ms.

Python program calls Scala UDF with function call (PSF)

```
def extract_domain_scala(col):
    _extract_domain_scala = sc._jvm.io.archivesunleashed.df.ExtractDomainStandAlone.getFun()
    return Column(_extract_domain_scala.apply(_to_seq(sc, [col], _to_java_column)))

data = pg
    .select(extract_domain_scala(col('Url')))
    .groupBy('UDF(Url)')
    .count()
    .orderBy("count", ascending=False)
    .head(3)
```

Expression used

```
d3=c(5587,5617,5815,5614,5802,5567,5654,5614,5577,5654)
m3=mean(d3)
s3=sd(d3)
```

The mean process time is 5650.1 ms.

Python program calls Scala UDF with SQL expression (PSS)

```
data = pg
    .selectExpr("extractDomainReg(Url) as Domain")
    .groupBy("Domain")
    .count()
    .orderBy("count", ascending=False)
    .head(3)
```

Expression used

```
d4=c(5674,5781,5984,5846,5841,5577,5893,5868,5941,5581)
m4=mean(d4)
s4=sd(d4)
```

The mean process time is 5798.6 ms.

Python program calls Python UDF with function call (PPF)

```
data = pg
    .withColumn('Domain', extract_domain(pg.Url))
    .groupBy("Domain")
    .count()
    .orderBy("count", ascending=False)
    .head(3)
```

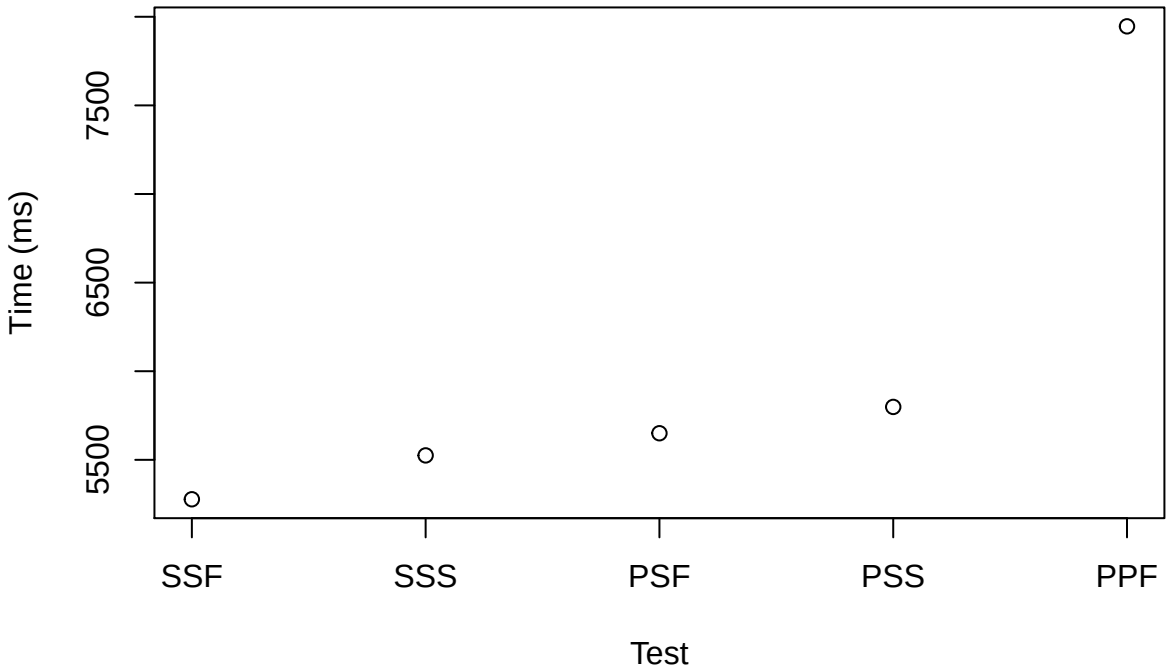
Expression used

```
d5=c(8041,7782,7835,8230,7796,7867,8134,7728,8071,7976)
m5=mean(d5)
s5=sd(d5)
```

The mean process time is 7946 ms.

Result

```
comp=c(m1,m2,m3,m4,m5)
n=c("Scala/Scala/Function",
    "Scala/Scala/SQL",
    "Python/Scala/Funcion",
    "Python/Scala/SQL",
    "Python/Python/Function")
n2=c("SSF", "SSS", "PSF", "PSS", "PPF")
plot(comp, main="Scala UDF vs Python UDF", ylab="Time (ms)", xlab="Test", xaxt="n")
axis(1, at=1:5, labels=n2)
```



From the graph, it can be shown that Scala UDFs, no matter where they were called, are always the fastest implementation comparing to the equivalent version in Python. It is also found that calling Scala UDF from Python does suffer from overhead of crossing language boundary, and this overhead is around ten to twenty percent. Also, calling registered UDFs in a SQL expression is slower both in Python and Scala, comparing to directly invoking UDFs in Scala or Python scripts, with 'select' method of data frame class. This difference is possibly due to the time it takes to parse and evaluate SQL expressions before they can be acted upon.

Conclusion

Frameworks that wish to provide UDFs to users should implement UDFs natively in Scala, and expose those functions to Python scripts. If used together with 'select' method of Data Frames in PySpark, the running time can be reduced to the smallest possible.

References

University of Toronto Libraries, Canadian Political Parties and Interest Groups, Archive-It Collection 227, Canadian Action Party, http://wayback.archive-it.org/227/20051004191340/ http://canadianactionparty.ca/Default2.asp

python udf vs python vectorized udf vs scala udf, https://gist.github.com/maropu/9f995f65b1cb160865e79e14e5216320

Spark User Defined Functions (UDFs), https://medium.com/@mrpowers/spark-user-defined-functions-udfs-6c849e39443b

UDFs—User-Defined Functions, https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-udfs.html

Spark: Custom UDF Example, https://ragrawal.wordpress.com/2015/10/02/spark-custom-udf-example/

How to Turn Python Functions into PySpark Functions (UDF), http://changhsinlee.com/pyspark-udf/

How to Use Scala UDF and UDAF in PySpark, http://www.cyanny.com/2017/09/15/spark-use-scala-udf-udaf-in-pyspark/

Appendix

- Web archive data used

archivesunleashed/aut-resources

- Github repository containing all tests

TitusAn/aut

- Git diff file towards archivesunleashed/aut/ef6ea3627128254a6eaa9a6022989545fb61da44 containing all tests.

git.diff