Yandex

Working with text files

Sample data

› Financial data from http://finance.yahoo.com for NASDAQ
› Stored in the CSV format in the file 'nasdaq.csv'

```
2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000
2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000
2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000
2017-01-06,5499.080078,5536.520020,5482.810059,5521.060059,5521.060059,1710770000
2017-01-09,5527.580078,5541.080078,5517.140137,5531.819824,5531.819824,1885500000
2017-01-10,5536.540039,5564.250000,5528.109863,5551.819824,5551.819824,1796500000
2017-01-11,5550.720215,5564.080078,5524.029785,5563.649902,5563.649902,1954720000
2017-01-12,5542.560059,5550.669922,5496.819824,5547.490234,5547.490234,1801750000
2017-01-13,5557.569824,5584.259766,5557.200195,5574.120117,5574.120117,1605110000
2017-01-17,5555.160156,5557.049805,5527.220215,5538.729980,5538.729980,1757030000
...
```

$ pyspark

```
...
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>> raw_data.take(3)
['2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000',
'2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000',
'2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000']
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>> raw_data.take(3)
['2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000',
'2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000',
'2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000']
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                       "adj_close", "volume"])
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>> raw_data.take(3)
['2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000',
'2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000',
'2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000']
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   # py3-specific; in py2 the last argument must be named
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>> raw_data.take(3)
['2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000',
'2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000',
'2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000']
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                    "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   # py3-specific; in py2 the last argument must be named
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = raw_data.map(parse_record)
>>> parsed_data.take(1)
[Record(date='2017-01-03', open=5425.620117, high=5452.569824,
   low=5397.990234, close=5429.080078, adj_close=5429.080078, volume=1886200000)]
>>>
```

$ pyspark

```
...
>>> raw_data = sc.textFile("nasdaq.csv")
>>> raw_data
nasdaq.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
>>> raw_data.take(3)
['2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000',
'2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000',
'2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000']
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   # py3-specific; in py2 the last argument must be named
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = raw_data.map(parse_record)
>>> parsed_data.take(1)
[Record(date='2017-01-03', open=5425.620117, high=5452.569824,
   low=5397.990234, close=5429.080078, adj_close=5429.080078, volume=1886200000)]
>>> parsed_data.cache()
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> parsed_data.map(lambda x: x.date).min()
'2017-01-03'
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                          "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> parsed_data.map(lambda x: x.date).min()
'2017-01-03'
>>> parsed_data.map(lambda x: x.date).max()
'2017-07-21'
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> parsed_data.map(lambda x: x.date).min()
'2017-01-03'
>>> parsed_data.map(lambda x: x.date).max()
'2017-07-21'
>>> parsed_data.map(lambda x: x.volume).sum()
265622040000
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> parsed_data.map(lambda x: x.date).min()
'2017-01-03'
>>> parsed_data.map(lambda x: x.date).max()
'2017-07-21'
>>> parsed_data.map(lambda x: x.volume).sum()
265622040000
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                    "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> with_month_data = parsed_data.map(lambda x: (x.date[:7], x))
>>> with_month_data.take(1)
[('2017-01', Record(...))]
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> with_month_data = parsed_data.map(lambda x: (x.date[:7], x))
>>> with_month_data.take(1)
[('2017-01', Record(...))]
>>> by_month_data = with_month_data.mapValues(lambda x: x.volume)
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> with_month_data = parsed_data.map(lambda x: (x.date[:7], x))
>>> with_month_data.take(1)
[('2017-01', Record(...))]
>>> by_month_data = with_month_data.mapValues(lambda x: x.volume)
>>> by_month_data = by_month_data.reduceByKey(lambda x, y: x + y)
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> with_month_data = parsed_data.map(lambda x: (x.date[:7], x))
>>> with_month_data.take(1)
[('2017-01', Record(...))]
>>> by_month_data = with_month_data.mapValues(lambda x: x.volume)
>>> by_month_data = by_month_data.reduceByKey(lambda x, y: x + y)
>>> by_month_data.top(1, lambda x: x[1])
[('2017-06', 48689910000)]
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> by_month_data = parsed_data.map(lambda x: (x.date[:7], x.volume)) \
                       .reduceByKey(lambda x, y: x + y)
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> by_month_data = parsed_data.map(lambda x: (x.date[:7], x.volume)) \
                    .reduceByKey(lambda x, y: x + y)
>>> result_data = by_month_data.map(lambda t: ",".join(map(str, t)))
>>> result_data.take(1)
['2017-03,43937720000']
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> by_month_data = parsed_data.map(lambda x: (x.date[:7], x.volume)) \
                    .reduceByKey(lambda x, y: x + y)
>>> result_data = by_month_data.map(lambda t: ",".join(map(str, t)))
>>> result_data.take(1)
['2017-03,43937720000']
>>> result_data.saveAsTextFile("out")
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> by_month_data = parsed_data.map(lambda x: (x.date[:7], x.volume)) \
                    .reduceByKey(lambda x, y: x + y)
>>> result_data = by_month_data.map(lambda t: ",".join(map(str, t)))
>>> result_data.take(1)
['2017-03,43937720000']
>>> result_data.saveAsTextFile("out")
>>> ^Z

$ ls out/
_SUCCESS   part-00000  part-00001
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> by_month_data = parsed_data.map(lambda x: (x.date[:7], x.volume)) \
                     .reduceByKey(lambda x, y: x + y)
>>> result_data = by_month_data.map(lambda t: ",".join(map(str, t)))
>>> result_data.take(1)
['2017-03,43937720000']
>>> result_data.repartition(1).saveAsTextFile("out")
>>>
```

Summary

- You have learned how to:
  - load and save text files from Pyspark
  - explore datasets
  - make keyed datasets and use keyed transformations and actions

BigDATAteam