Assignment 5

Code to realize MapReduce

The MapReduce programming style was stirred by the functional programming constructs map and reduce. Those are usually used to process lists of data. Each Map-Reduce program converts a list of input data elements into a list of output data elements twice at a high level. That is transformed once in the map phase and once in the reduce phase.

The MapReduce structure is composed of three major phases.

Map

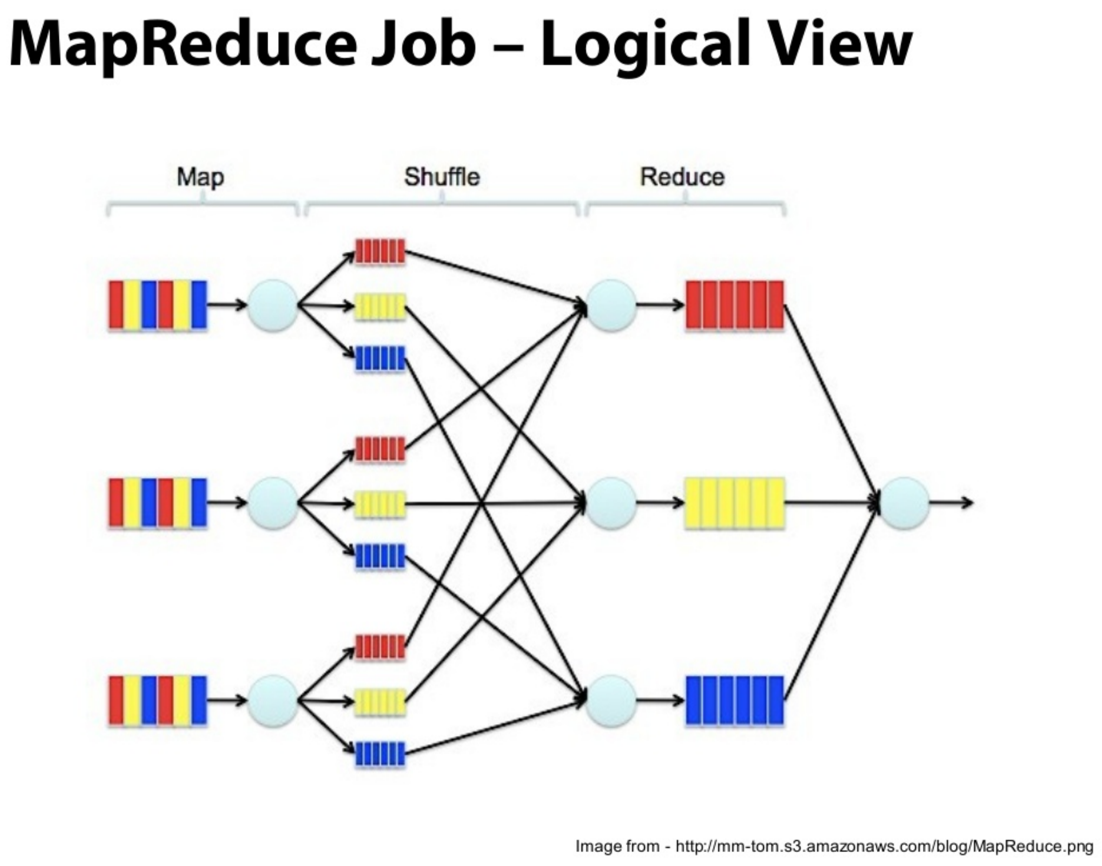
The first stage of a MapReduce application is the map stage.A function that is called the mapper, routes a series of key-value pairs inside the map stage.The mapper serially processes every key-value pair separately, creating zero or more output key-value pairs.

Shuffle and Sort

The second stage of MapReduce is the shuffle and sort.The intermediate outputs from the map stage are moved to the reducers as the mappers bring into being completing.This process of moving output from the mappers to the reducers is recognized as shuffling.Shuffling is moved by a divider function, named the partitioner.The partitioner is used to handle the flow of key-value pairs from mappers to reducers.The partitioner is provided the mapper’s output key and the number of reducers.

Reduce

The third stage of MapReduce is the reduce stage.An iterator of values is given to a function known as the reducer inside the reducer phase.The iterator of values is a non-single set of values for every unique key from the output of the map stage.The reducer sums the values for every single key and yields zero or more output key-value pairs.


MapReduce Job – Logical View
Map
Shuffle
Reduce
Image from - http://mm-tom.s3.amazonaws.com/blog/MapReduce.png

MAP PHASE

Python provides map function which maps directly the given input.

In [33]:
```
num= [1,3.4,5,6,4.2,21.1,12,3,3.8,7.6,3.5,4.7,12.4,2.34,5.5]
mapped=list(map(lambda x:x*x,num))
mapped
```

Out[33]:
```
[1,
 11.559999999999999,
 25,
 36,
 17.64,
 445.21000000000004,
 144,
 9,
 14.44,
 57.76,
 12.25,
 22.090000000000003,
 153.76000000000002,
 5.475599999999999,
 30.25]
```

In above code we have an input set and map function is mapping it to square of inputs,

SUFFLE & SORT

In [34]:
```
## Filtering value that is greater than 10
def check(a):
    return a>10
filtered=list(filter(check,mapped))
filtered.sort()
```

In [35]:
```
filtered ##sorted out values greater than 10
```

Out[35]:
```
[11.559999999999999,
 12.25,
 14.44,
 17.64,
 22.090000000000003,
 25,
 30.25,
 36,
 57.76,
 144,
 153.76000000000002,
 445.21000000000004]
```

REDUCE

In [36]:
```
# reducing the filtered function as their sum
from functools import reduce
red=reduce(lambda x,y:x+y,mapped)
```

In [37]:
```
red
```

Out[37]:
```
985.4356000000001
```

In [47]:
```
# Another example of using map to find square root of
# given data after suffle and reduce to sum of their value using map and
import math
num= [1,3,5,6,4,21,12,3,3,7,3.5,4.7,12.4,2.34,5.5]
mapped=list(map(math.sqrt,num))
def check(a):
    return a>3
filtered=list(filter(check,mapped))
filtered.sort()
red=reduce(lambda x,y:x+y,mapped)
```

In [51]:
```
num #input
```

Out[51]:
```
[1, 3, 5, 6, 4, 21, 12, 3, 3, 7, 3.5, 4.7, 12.4, 2.34, 5.5]
```

```
In [48]: mapped # mapped value using map function
```

```
Out[48]: [1.0,
          1.7320508075688772,
          2.23606797749979,
          2.449489742783178,
          2.0,
          4.58257569495584,
          3.4641016151377544,
          1.7320508075688772,
          1.7320508075688772,
          2.6457513110645907,
          1.8708286933869707,
          2.16794833886788,
          3.521363372331802,
          1.5297058540778354,
          2.345207879911715]
```

```
In [52]: red #reduces value
```

```
Out[52]: 35.00919290272398
```

```
In [ ]:
```