Hands on with Apache Spark Computing Framework

slides link:
http://bit.ly/1BQNDg5

Speaker: Charlie, a programmer at 

https://www.udomain.hk/

Installation

Have JDK 7 installed

Then, follow the documentation at:

https://spark.apache.org/docs/latest/

For this workshop:

- Use the vm we provide for you.

Today's Objectives

- Understand the basic idea of Spark
- Open a Spark Shell
- Write one line code in Spark Shell
- Use simple transformations
- Use simple actions
- Submit your code to Spark Master
- Connect Spark in your own app

Basic Ideas

- SparkContext
  - Main entry point.
  - The connection cluster
  - Can be used to create partitioned datasets
  - Can have accumulators
  - Can broadcast variables to cluster nodes.

Basic Ideas

- RDDs (Resilient Distributed Datasets
  - Abstraction of the data distributed across the cluster.
  - Kept in memory (better than disk I/O)
  - Transformations
  - Actions

Step 1: Run the shell

To use Scala:

```
./bin/spark-shell
```

More about Scala programming, See: http://scala-lang.org

To use Python:

```
./bin/pyspark
```

Then, from the “>>>” prompt,create some data:

```
data = range(1, 10000)
```

Step 1: Prepare the data

```
cd /home/vagrant/spark/101workshop
```

Get a error log file here:

wget https://raw.githubusercontent.com/Carolusian/hkoscon2015/master/error_log

Step 2: Run the shell in ipython notebook

Run the same shell in ipython-notebook:

```
cd /home/vagrant/spark/101workshop
export SPARK_HOME=/home/vagrant/spark
ipython notebook --profile=pyspark
```

Then, open you browser on your laptop, open:

http://127.0.0.1:8089

Step 3: Create a RDD (resilient distributed datasets)

Write the following code:

```
data = range(1, 10000)
```

```
dist_data = sc.parallelize(data)
```

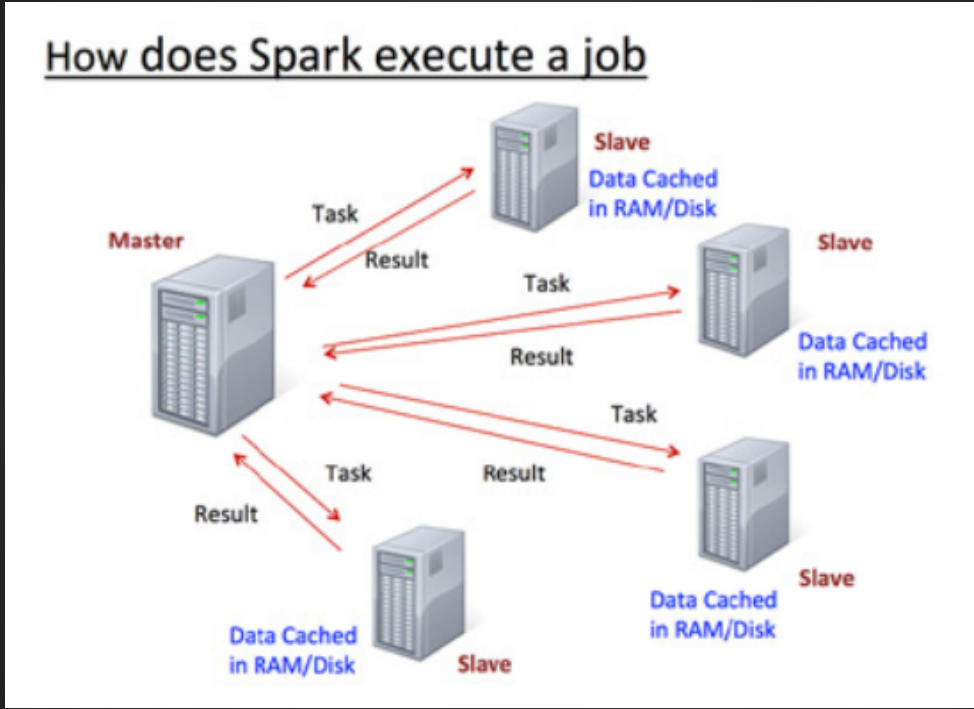
Then, we can select integers less than sth:

```
dist_data.filter(lambda i: i < 100).collect()
```

Step 4: Error log example

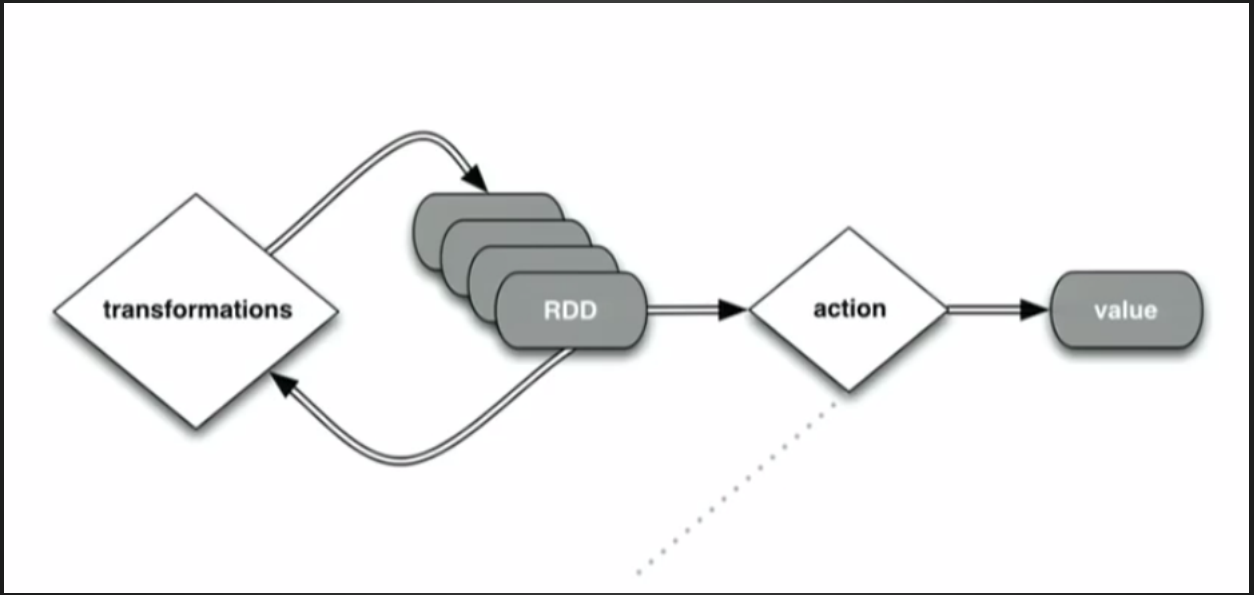
```
# Load the apache error_log into memory
# Then interactively search for patterns that are useful for engineers.

# base RDD, can be from other storage resources like hdfs and amazon s3
lines = sc.textFile("hdfs://[url for a batch of log files]")

# transformed RDDs
errors = lines.filter(lambda l: "error" in l)

# persistence
errors.cache()
errors.filter(lambda e: e.find("61.9.4.61") > -1).count() # action 1
errors.filter(lambda e: e.find("200.174.151.3") > -1).count() # action 2
```

Step 4: Error log example



From: http://www.mapr.com/

Step 4: Error log example



From Spark Summit 2014

Step 5: common transformations

Transformation	Example

`filter(func)`
Purpose: get a new RDD in accordance with the given func which returns true
	`rdd = sc.parallelize(["ABC", "BCD", "DEF"])`
	`filtered = rdd.filter(lambda x: "C" in x)`
	`filtered.collect()`

`map(func)`
`Purpose: return new RDD by applying func on each element`
	`rdd = sc.parallelize([1,2,3,4,5])`
	`times2 = rdd.map(lambda x: x * 2)`
	`times2.collect()`

`flatMap(func)`
`Purpose: similar to map, but func return sequence`
	`rdd = sc.parallelize(["Spark is awesome", "It is fun"])`
	`fm = rdd.flatMap(lambda x: x.split(" "))`
	`fm.collect()`

Step 5: common transformations

Transformation	Example
`reduceByKey(func)` Purpose: aggregate values of key.	`word1 = fm.map(lambda word: (word, 1))` `word_count = word1.reduceByKey(lambda x,y: x + y).collect()`
`groupByKey()` `Purpose: to convert (K,V) to (K, Iteratable<V>)`	`cnt_w = word_count.map(lambda (w, cnt): (cnt, w))` `cnt_w.groupByKey().collect()`
`distinct()` `Purpose: eliminate duplicated elements in RDD`	`fm.distinct().collect()`

See: https://spark.apache.org/docs/latest/programming-guide.html#transformations

Step 5: common actions

Action	Example
count() Purpose: get the number of data elements in RDD	`rdd = sc.parallelize(['A','B','C'])` `rdd.count()`
collect() Purpose: get all the data elements in RDD	`rdd = sc.parallelize(['A','B','C'])` `rdd.collect()`
reduce(func) Purpose: eliminate duplicated elements in RDD	`fm.distinct().collect()`
take(n) Purpose: fetch first n data elements	`rdd = sc.parallelize(['A','B','C'])` `rdd.take(2)`

See: https://spark.apache.org/docs/latest/programming-guide.html#actions

Step 6: Wordcount - the Hello World sample in Big Data

```
log = sc.textFile("/home/vagrant/spark/101workshop/error_log")
# Split lines into words
word_cnt = log.flatMap(lambda line: line.split(" ")) \
    # Get rid of empty words
    .filter(lambda w: bool(w)) \
    # Each word appear once
    .map(lambda w: (w, 1)) \
    # For the same word, count the times it appears
    .reduceByKey(lambda wc1, wc2: wc1 + wc2) \
    # Switch the count and the word in the tuple
    .map(lambda w: (w[1], w[0])) \
    # Descendant sort in accordance with the count
    .sortByKey(False)

# Logic graph of the transformations
word_cnt.toDebugString()
word_cnt.take(5)
word_cnt.saveAsTextFile("/home/vagrant/spark/101workshop/word_count.txt")
```

Step 7: Submit your code to cluster - run the cluster

```
cd /home/vagrant/spark
```

```
./sbin/start-master.sh
```

```
Open your browser at: http://127.0.0.1:8081
```

Step 7: Submit your code to cluster - submit sample code

```
# Run on a Spark Standalone cluster in client deploy mode
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master spark://master:7077 \
  --executor-memory 20G \
  --total-executor-cores 100 \
  ~/spark/lib/spark-examples-1.3.1-hadoop2.6.0.jar \
  1000
```

Step 7: Submit your code to cluster - start worker node

```
# You will find the job is waiting for execution
# So you need worker node to execute the jobs
# After worker node started, you will find the job running

./bin/spark-class org.apache.spark.deploy.worker.Worker spark://master:7077
```

Step 7: Submit your code to cluster - you own code

```
# put it at /home/vagrant/spark/101workshop/simplefilter.py
from pyspark import SparkContext

if __name__ == "__main__":

    sc = SparkContext(appName="Simple Filter")

    numbers = sc.parallelize(range(1, 10000))
    even = numbers.filter(lambda n: n % 2 == 0)
    print even.take(5)
```

Step 7: Submit your code to cluster - exercise/homework

```
# Create a wordcount.py program

# Submit it
```

```
>>> Thank You !
```