



# EECS E6893 Big Data Analytics

## Spark 101

Frank Ou Yang, [ho2271@columbia.edu](mailto:ho2271@columbia.edu)

# Agenda

- More on GCP Cloud Shell
- Functional programming in Python
  - Lambda
- Crash course in Spark (PySpark)
  - RDD
  - Useful RDD operations
    - Actions
    - Transformations
  - Example: Word count



# GCP Cloud Shell

# GCP: Interaction

- [Graphical UI / console](#): Useful to create VMs, set up clusters, provision resources, manage teams, etc
- [Command line tools / Cloud SDK](#): Useful for interacting from local host and using the resources once provisioned. E.x. ssh into instances, submit jobs, copy files, etc
- [Cloud shell](#): Same as command line, but web-based and pre-installed with SDK and tools

# GCP: Cloud Shell



The screenshot shows the Google Cloud Platform dashboard for the project 'big-data-ta'. The top navigation bar is blue and contains the Google Cloud Platform logo, the project name 'big-data-ta', a search bar, and a user profile icon. A red rectangle highlights the 'Activate Cloud Shell' button, which is located next to the user profile icon. Below the navigation bar, the dashboard is divided into three main sections: 'Project info', 'API APIs', and 'Google Cloud Platform status'. The 'Project info' section on the left displays the project name 'big-data-ta', Project ID 'logical-host-251101', and Project number '312759131343'. The 'API APIs' section in the middle shows a line graph of 'Requests (requests/sec)' with a peak of approximately 2.2. The 'Google Cloud Platform status' section on the right indicates 'All services normal' and provides a link to the 'Cloud status dashboard'. Below this, the 'Billing' section shows 'Estimated charges' of 'USD \$0.00' for the billing period 'Sep 1 - 12, 2019'.

Google Cloud Platform big-data-ta

DASHBOARD ACTIVITY

Activate Cloud Shell CUSTOMIZE

**Project info**

- Project name: big-data-ta
- Project ID: logical-host-251101
- Project number: 312759131343

→ Go to project settings

**API APIs**

Requests (requests/sec)

Google Cloud Platform status

All services normal

→ Go to Cloud status dashboard

**Billing**

Estimated charges: USD \$0.00

For the billing period Sep 1 - 12, 2019

persistent home directory :)

# GCP: Cloud Shell (Cont')

The screenshot displays the Google Cloud Platform (GCP) dashboard for the project 'big-data-ta'. The top navigation bar includes the GCP logo, the project name, a search bar, and user profile icons. The main content area is divided into three panels: 'Project info', 'API APIs', and 'Google Cloud Platform status'. The 'Project info' panel lists the project name 'big-data-ta', project ID 'logical-host-251101', and project number. The 'API APIs' panel shows a line graph for 'Requests (requests/sec)' with a peak around 2.5. The 'Google Cloud Platform status' panel indicates 'All services normal' and provides a link to 'Go to Cloud status dashboard'. At the bottom, a terminal window shows the Cloud Shell environment with the prompt 'frouyang2@cloudshell:~\$' and the command 'ls' executed, displaying 'hw0 README-cloudshell.txt'. A 'Launch code editor BETA' button is visible in the bottom right corner.

Google Cloud Platform big-data-ta

DASHBOARD ACTIVITY CUSTOMIZE

**Project info**

- Project name: big-data-ta
- Project ID: logical-host-251101
- Project number

**API APIs**

Requests (requests/sec)

2.5

2.0

**Google Cloud Platform status**

All services normal

→ Go to Cloud status dashboard

cloudshell x +

```
frouyang2@cloudshell:~$ ls
hw0 README-cloudshell.txt
frouyang2@cloudshell:~$
```

Launch code editor BETA

# GCP: Code Editor

The screenshot displays the Google Cloud Shell environment. At the top, the title bar reads "Cloud Shell" with standard window controls. Below it is a menu bar with "File", "Edit", "Selection", "View", "Go", and "Help". The interface is divided into three main sections: an Explorer on the left, a file browser in the center, and a code editor on the right.

The Explorer on the left shows a directory structure starting with "FROUYANG2", which contains a subdirectory "hw0". Inside "hw0", the file "wordcount.py" is selected and highlighted in blue.

The central file browser shows the selected file "wordcount.py" with a tab icon and the filename.

The code editor on the right displays the contents of "wordcount.py". The code is as follows:

```
1  #!/usr/bin/env python
2
3  import pyspark
4  import sys
5  import nltk
6  nltk.download('stopwords')
7  from nltk.corpus import stopwords
8
9  stopwords = set(stopwords.words('english'))
10 print(stopwords)
11
12 inputUri = "gs://big_data_ta/input/rose.txt"
13
14 sc = pyspark.SparkContext()
15
16 lines = sc.textFile(inputUri)
17 words = lines.flatMap(lambda line: line.split())
```

At the bottom of the interface is a terminal window. It shows the prompt "frouyang2@cloudshell:~/hw0\$ ls" followed by the output "wordcount.py". The next prompt "frouyang2@cloudshell:~/hw0\$" is shown with a cursor, indicating the user is ready to enter a command.

# Functional programming in Python



# Lambda expression

- Creating small, one-time, anonymous function objects in Python
- **Syntax:** `lambda arguments: expression`
  - Any number of arguments
  - Single expression
- Could be used together with *map*, *filter*, *reduce*
- **Example:**
  - Add:

```
add = lambda x, y : x + y
```

```
type(add) = <type 'function'>
```

```
add(2,3)
```

```
def add (x, y):  
    return x + y
```



# Crash course in Spark

# Resilient Distributed Datasets (RDD)

- *An abstraction*
  - a collection of elements
  - partitioned across the nodes of the cluster
  - can be operated on in parallel
- Spark is RDD-centric
- RDDs are immutable
- RDDs can be cached in memory
- RDDs are computed lazily
- RDDs know who their parents are
- RDDs automatically recover from failures

# Useful RDD Actions

- `take(n)`: return the first `n` elements in the RDD as an array.
- `collect()`: return all elements of the RDD as an array. Use with caution.
- `count()`: return the number of elements in the RDD as an int.
- `saveAsTextFile('path/to/dir')`: save the RDD to files in a directory. Will create the directory if it doesn't exist and will fail if it does.
- `foreach(func)`: execute the function against every element in the RDD, but don't keep any results.

# Useful RDD transformations

# map(*func*)

- Apply a function to every element of an RDD and return a new result RDD

```
data = ["Apple,Amy", "Butter,Bob", "Cheese,Chucky"]  
data = sc.parallelize(data)
```

```
# map  
data.map(lambda line: line.split(',')).take(3)  
  
[['Apple', 'Amy'], ['Butter', 'Bob'], ['Cheese', 'Chucky']]
```

```
data.map(lambda line: line.lower()).take(3)  
  
['apple,amy', 'butter,bob', 'cheese,chucky']
```

# flatMap(*func*)

- Similar to *map()*, yet flatten by removing the outermost container

```
# flatMap  
data.flatMap(lambda line: line.split(',')).take(6)
```

```
['Apple', 'Amy', 'Butter', 'Bob', 'Cheese', 'Chucky']
```

# mapValues(*func*)

- Apply an operation to the value of every element of an RDD and return a new result RDD
- Only works with pair RDDs

```
pair_data = [('Apple', 'Amy'), ('Butter', 'Bob'), ('Cheese', 'Chucky')]  
pair_data = sc.parallelize(pair_data)
```

```
# mapValues()  
# each pair: (key, value)  
pair_data.mapValues(lambda name: name.lower()).take(3)
```

```
[('Apple', 'amy'), ('Butter', 'bob'), ('Cheese', 'chucky')]
```



# flatMapValues(*func*)

- Pass each value in the (K, V) pair RDD through a *flatMap* function without changing the keys

```
# flatMapValues()  
pair_data.flatMapValues(lambda name: name.lower()).take(6)
```

```
[('Apple', 'a'),  
 ('Apple', 'm'),  
 ('Apple', 'y'),  
 ('Butter', 'b'),  
 ('Butter', 'o'),  
 ('Butter', 'b')]
```

# filter(*func*)

- Return a new RDD by selecting the elements which func returns true

```
# filter
data = sc.parallelize([1, 2, 3, 4, 5])
data.filter(lambda x: x % 2 != 0).take(3)
```

```
[1, 3, 5]
```

# groupByKey()

- When called on a RDD of (K, V) pairs, returns a new RDD of (K, Iterable<V>) pairs

```
# groupByKey()
data = sc.parallelize([('A', 1), ('A', 2), ('B', 3), ('C', 4)])
print(data.groupByKey().take(1))

for pair in data.groupByKey().take(1):
    print(pair[0], [n for n in pair[1]])

[('A', <pyspark.resultiterable.ResultIterable object at 0x7f0b00a85290>)]
('A', [1, 2])
```

# reduceByKey(*func*)

- Combine elements of an RDD by key and then apply a *reduce func* to pairs of values until only a single value remains
- reduce function *func* must be of type  $(V,V) \Rightarrow V$

```
# reduceByKey()  
data = sc.parallelize([('A', 1), ('A', 2), ('B', 3), ('C', 4)])  
data.reduceByKey(lambda v1, v2: v1 + v2).take(1)
```

```
[('A', 3)]
```

# sortBy(*func*)

- Sort an RDD according to a sorting *func* and return the results in a new RDD

```
# sortBy()
data = sc.parallelize([('A', 99), ('B', 3), ('C', 4)])

print(data.sortBy(lambda pair: pair[1]).take(4))
print(data.sortBy(lambda pair: -pair[1]).take(4))
print(data.sortBy(lambda pair: pair[0]).take(4))
```

```
[('B', 3), ('C', 4), ('A', 99)]
[('A', 99), ('C', 4), ('B', 3)]
[('A', 99), ('B', 3), ('C', 4)]
```

# sortByKey()

- Sort an RDD according to the ordering of the keys and return the results in a new RDD.

```
# sortByKey()  
data = sc.parallelize([('A', 99), ('B', 3), ('C', 4)])  
data.sortByKey().take(3)
```

```
[('A', 99), ('B', 3), ('C', 4)]
```

# subtract()

- Return a new RDD that contains all the elements from the original RDD that do not appear in a target RDD.

```
# subtract
data1 = sc.parallelize(['Apple,Amy', 'Butter,Bob', 'Cheese,Chucky'])
data2 = sc.parallelize(['Wendy', 'McDonald,Ronald', 'Cheese,Chucky'])
data1.subtract(data2).take(3)

['Butter,Bob', 'Apple,Amy']
```

# Example: word count in Spark

- “Hello world” of Spark

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```



# Word count in Spark: read file into RDD (1)

```
text_file = sc.textFile("gs://big_data_ta/data/shakes.txt")
text_file.take(10)
```

```
[u"***The Project Gutenberg's Etext of Shakespeare's First Folio***",
 u'*****The Tragedie of Macbeth*****',
 u'',
 u'This is our 3rd edition of most of these plays.  See the index.',
 u'',
 u'',
 u'Copyright laws are changing all over the world, be sure to check',
 u'the copyright laws for your country before posting these files!!',
 u'',
 u'Please take a look at the important information in this header.']
```

## Word count in Spark: split into words (2)

```
words = text_file.flatMap(lambda line: line.split(" ")).filter(lambda x: x != ' ')  
words.take(10)
```

```
[u'***The',  
 u'Project',  
 u"Gutenberg's",  
 u'Etext',  
 u'of',  
 u"Shakespeare's",  
 u'First',  
 u'Folio***',  
 u'*****The',  
 u'Tragedie']
```

# Word count in Spark: form (k, v) pairs (3)

```
word_pairs = words.map(lambda x: (x, 1))  
word_pairs.take(10)
```

```
[(u'***The', 1),  
 (u'Project', 1),  
 (u"Gutenberg's", 1),  
 (u'Etext', 1),  
 (u'of', 1),  
 (u"Shakespeare's", 1),  
 (u'First', 1),  
 (u'Folio***', 1),  
 (u'*****The', 1),  
 (u'Tragedie', 1)]
```

# Word count in Spark: reduce by aggregating (4)

```
word_pairs.reduceByKey(lambda a, b: a + b).take(10)
```

```
[(u'bidding', 1),  
 (u'Lead', 1),  
 (u'hart,', 1),  
 (u'ever!', 1),  
 (u'wracke,', 2),  
 (u'protest', 1),  
 (u'Barke', 1),  
 (u'hate', 2),  
 (u"knoll'd", 1),  
 (u'grace,', 1)]
```

```
word_pairs.reduceByKey(lambda a, b: a + b).sortBy(lambda pair: -pair[1]).take(10)
```

```
[(u'the', 620),  
 (u'and', 427),  
 (u'of', 396),  
 (u'to', 367),  
 (u'I', 326),  
 (u'a', 256),  
 (u'you', 193),  
 (u'in', 190),  
 (u'is', 185),  
 (u'my', 170)]
```

# Next week tutorial

- Spark Dataframe and Spark SQL
- Spark MLlib
- HW1

# References

- GCP Cloud Shell
  - <https://cloud.google.com/shell/docs/quickstart>
- Python functional programming
  - [https://book.pythontips.com/en/latest/map\\_filter.html](https://book.pythontips.com/en/latest/map_filter.html)
  - <https://medium.com/better-programming/lambda-map-and-filter-in-python-4935f248593>
- Spark
  - RDD programming guide: <https://spark.apache.org/docs/latest/rdd-programming-guide.html>
  - Spark paper: [https://www.usenix.org/legacy/event/hotcloud10/tech/full\\_papers/Zaharia.pdf](https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/Zaharia.pdf)
  - RDD paper: <https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf>