CSCE 5300

Spark Streaming

What is Streaming?

- Data Streaming is a technique for transferring data so that it can be processed as a steady and continuous stream

- Streaming technologies are becoming increasingly important with the growth of the Internet

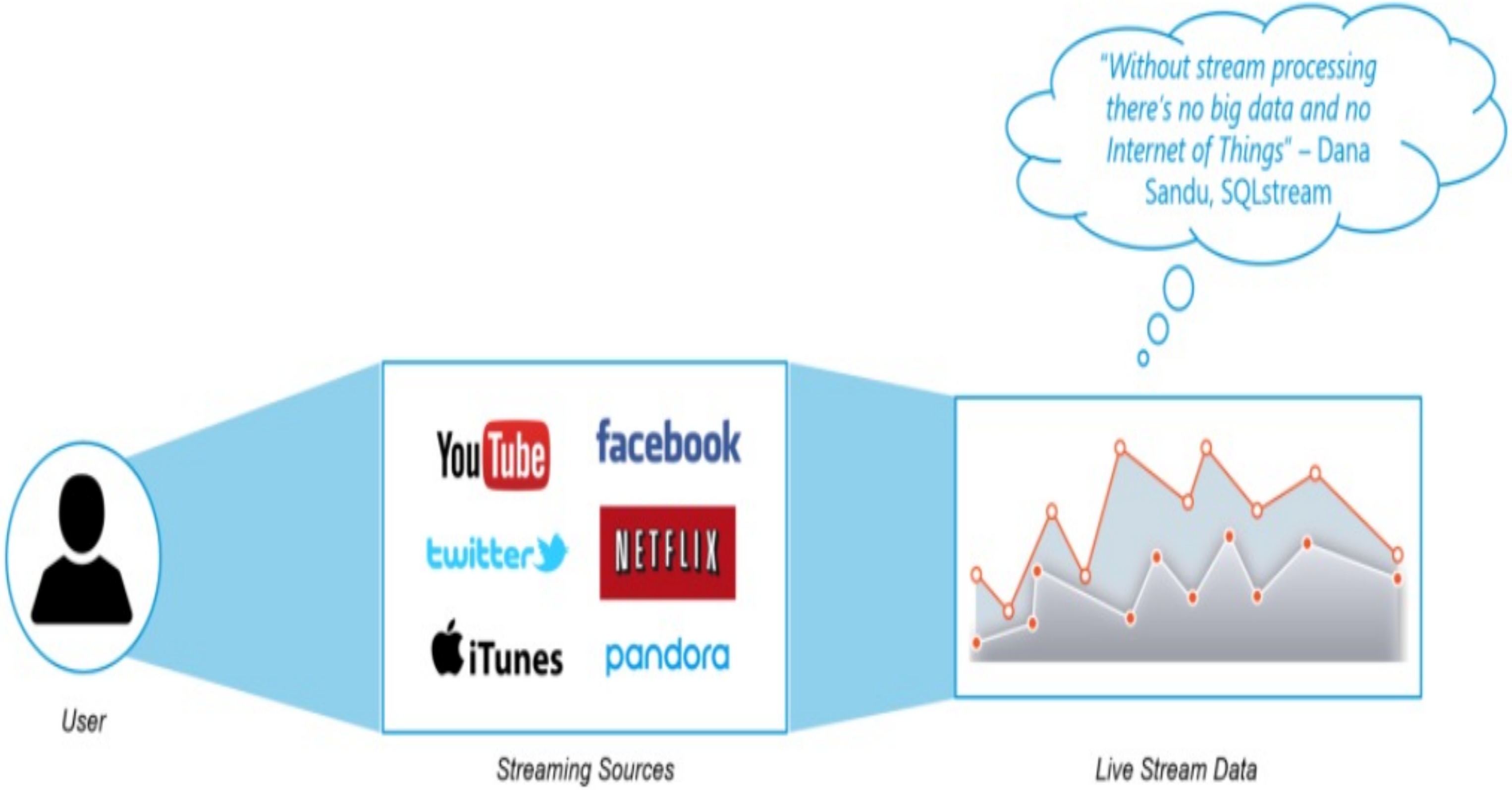
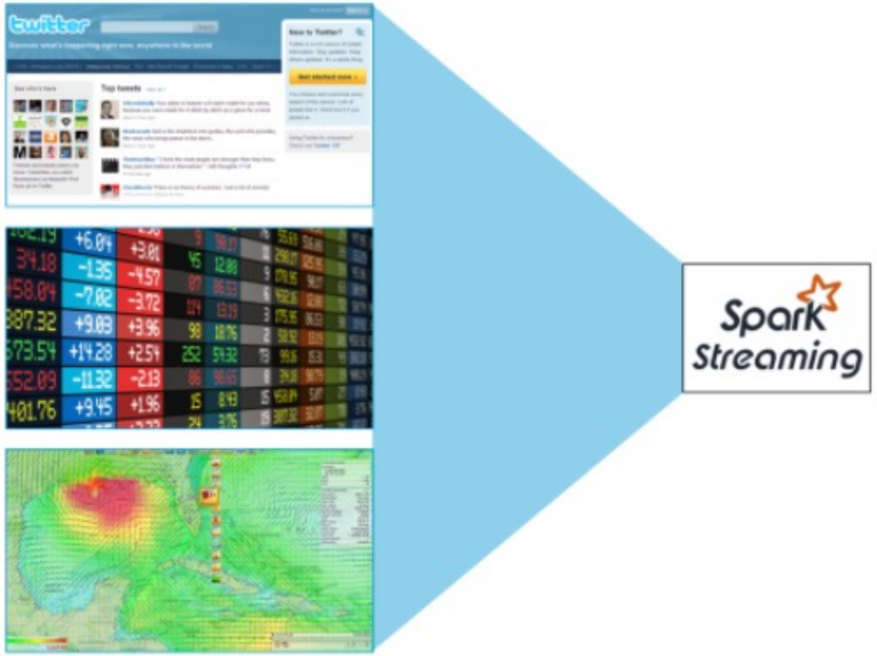

"Without stream processing there's no big data and no Internet of Things" – Dana Sandu, SQLstream
YouTube
facebook
twitter
NETFLIX
iTunes
pandora
User
Streaming Sources
Live Stream Data

Figure: *What is Streaming?*

Why Spark Streaming?



Spark Streaming is used to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.

Figure: *Why Spark Streaming?*

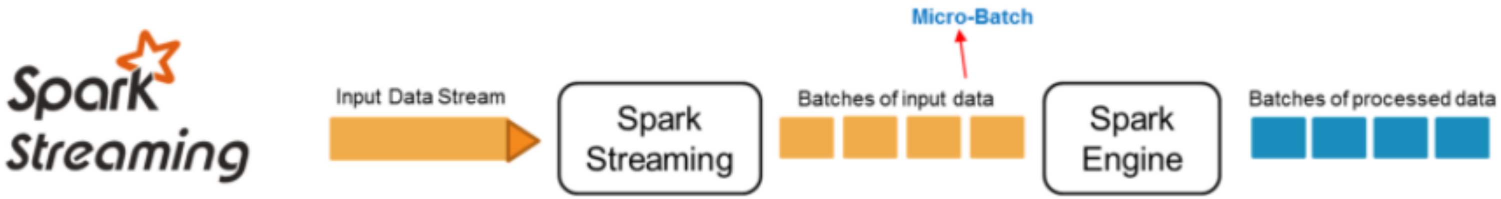


Figure: *Streams in Spark Streaming*

Spark Streaming Features

- **Scaling:** Spark Streaming can easily scale to hundreds of nodes.
- **Speed:** It achieves low latency.
- **Fault Tolerance:** Spark has the ability to efficiently recover from failures.
- **Integration:** Spark integrates with batch and real-time processing.
- **Business Analysis:** Spark Streaming is used to track the behavior of customers which can be used in business analysis

Spark Streaming Workflow

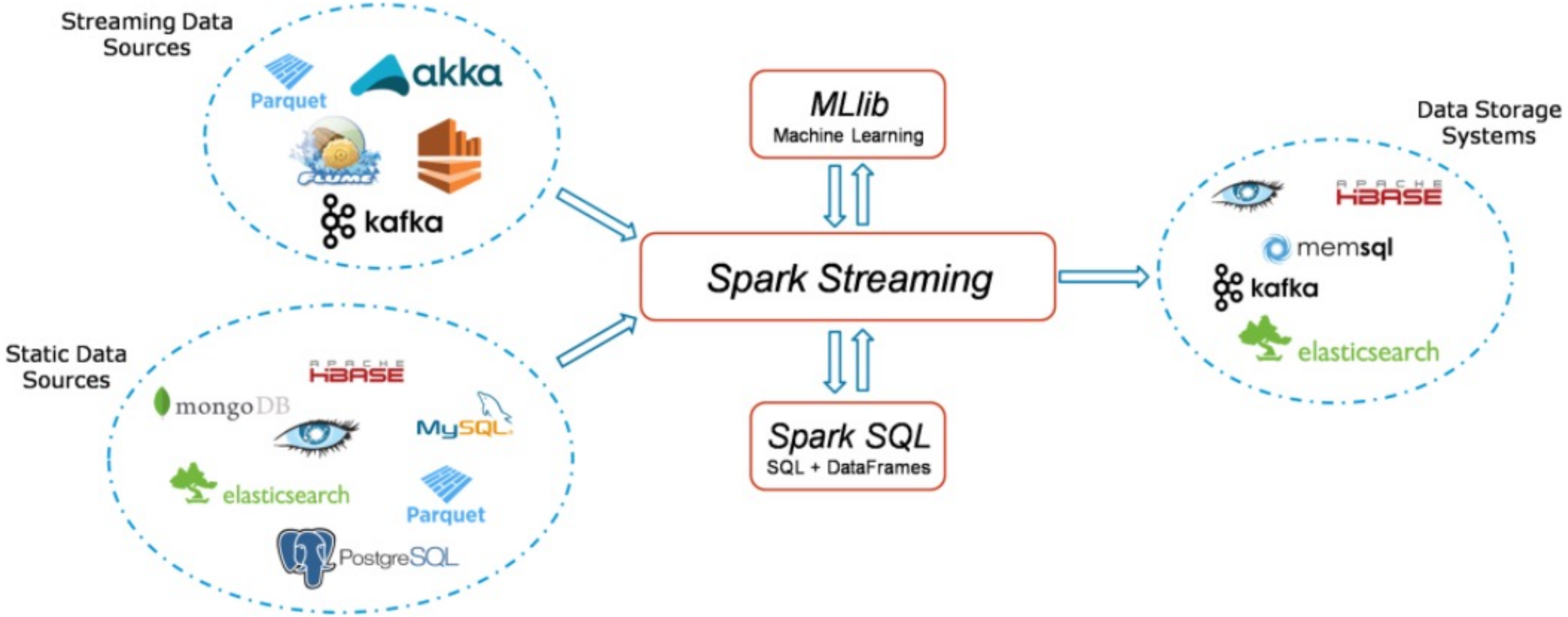


Figure: *Overview Of Spark Streaming*

Spark Streaming Fundamentals

- Streaming Context
- DStream
- Caching
- Accumulators, Broadcast Variables and Checkpoints

Streaming Context

- *Streaming Context* consumes a stream of data in Spark.
- It registers an *Input DStream* to produce a *Receiver* object.
- It is the main entry point for Spark functionality.
- Spark provides a number of default implementations of sources like Twitter, Akka Actor and ZeroMQ that are accessible from the context.

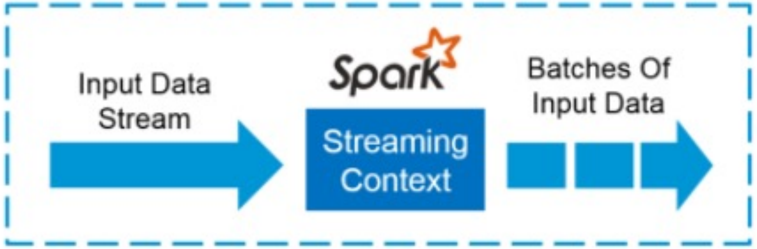


Figure: *Spark Streaming Context*

Figure: *Default Implementation Sources*

DStream

- *Discretized Stream* (DStream) is the basic abstraction provided by Spark Streaming.
- It is a continuous stream of data.
- It is received from a data source or a processed data stream generated by transforming the input stream.

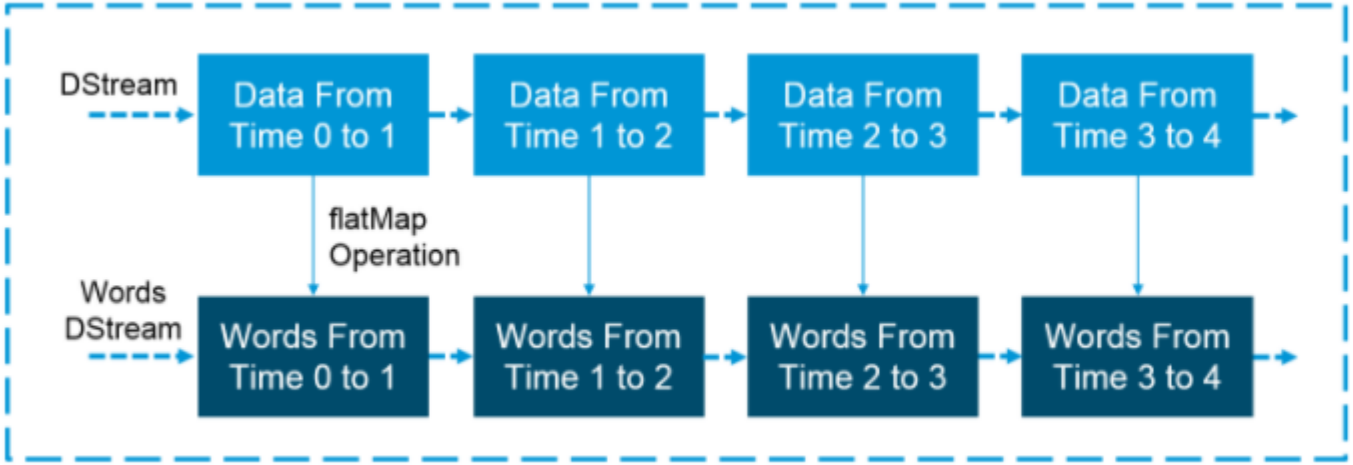
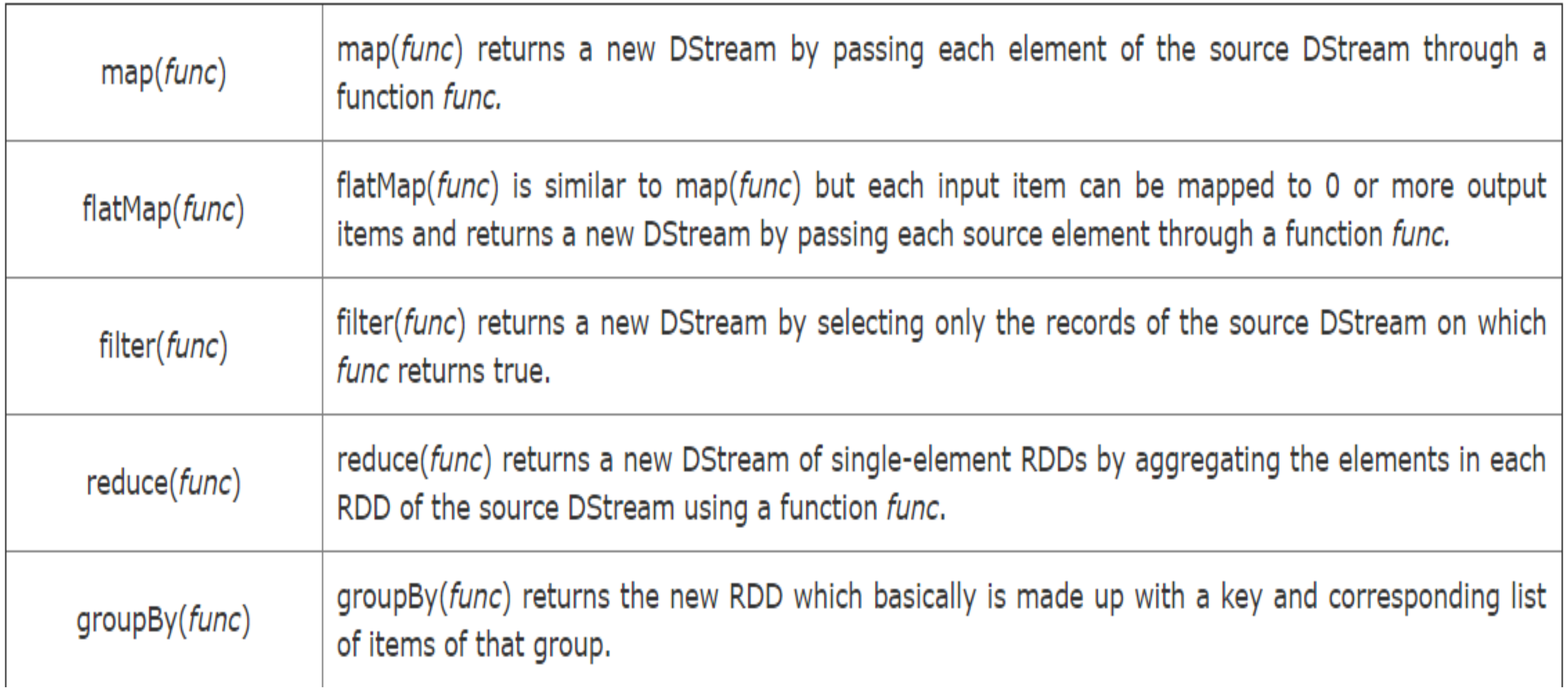


Figure: *Extracting words from an Input DStream*

Transformations on DStreams



map(*func*)	map(*func*) returns a new DStream by passing each element of the source DStream through a function *func*.
flatMap(*func*)	flatMap(*func*) is similar to map(*func*) but each input item can be mapped to 0 or more output items and returns a new DStream by passing each source element through a function *func*.
filter(*func*)	filter(*func*) returns a new DStream by selecting only the records of the source DStream on which *func* returns true.
reduce(*func*)	reduce(*func*) returns a new DStream of single-element RDDs by aggregating the elements in each RDD of the source DStream using a function *func*.
groupBy(*func*)	groupBy(*func*) returns the new RDD which basically is made up with a key and corresponding list of items of that group.

Accumulators, Broadcast Variables and Checkpoints

- **Accumulators:**

➢*Accumulators* are variables that are only added through an associative and commutative operation.

➢They are used to implement counters or sums. Tracking accumulators in the UI can be useful for understanding the progress of running stages.

➢Spark natively supports numeric accumulators. We can create named or unnamed accumulators.

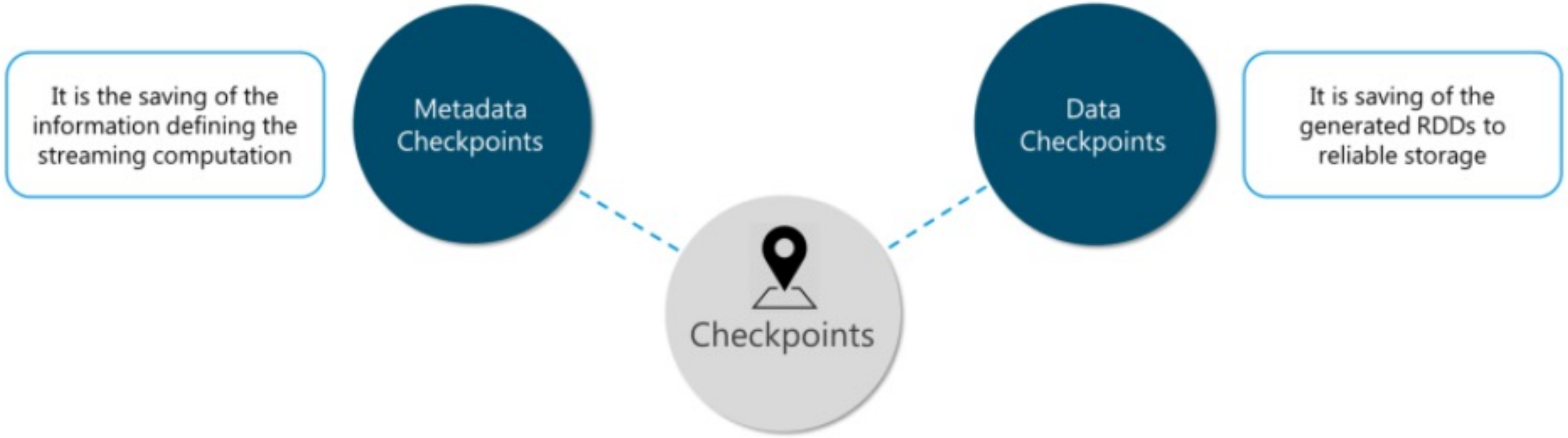
- **Broadcast Variables:**

➢ *Broadcast variables* allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks.

➢ They can be used to give every node a copy of a large input dataset in an efficient manner.

➢ Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost.

- **Checkpoints:** *Checkpoints* are similar to checkpoints in gaming. They make it run 24/7 and make it resilient to failures unrelated to the application logic.



It is the saving of the information defining the streaming computation

Metadata Checkpoints

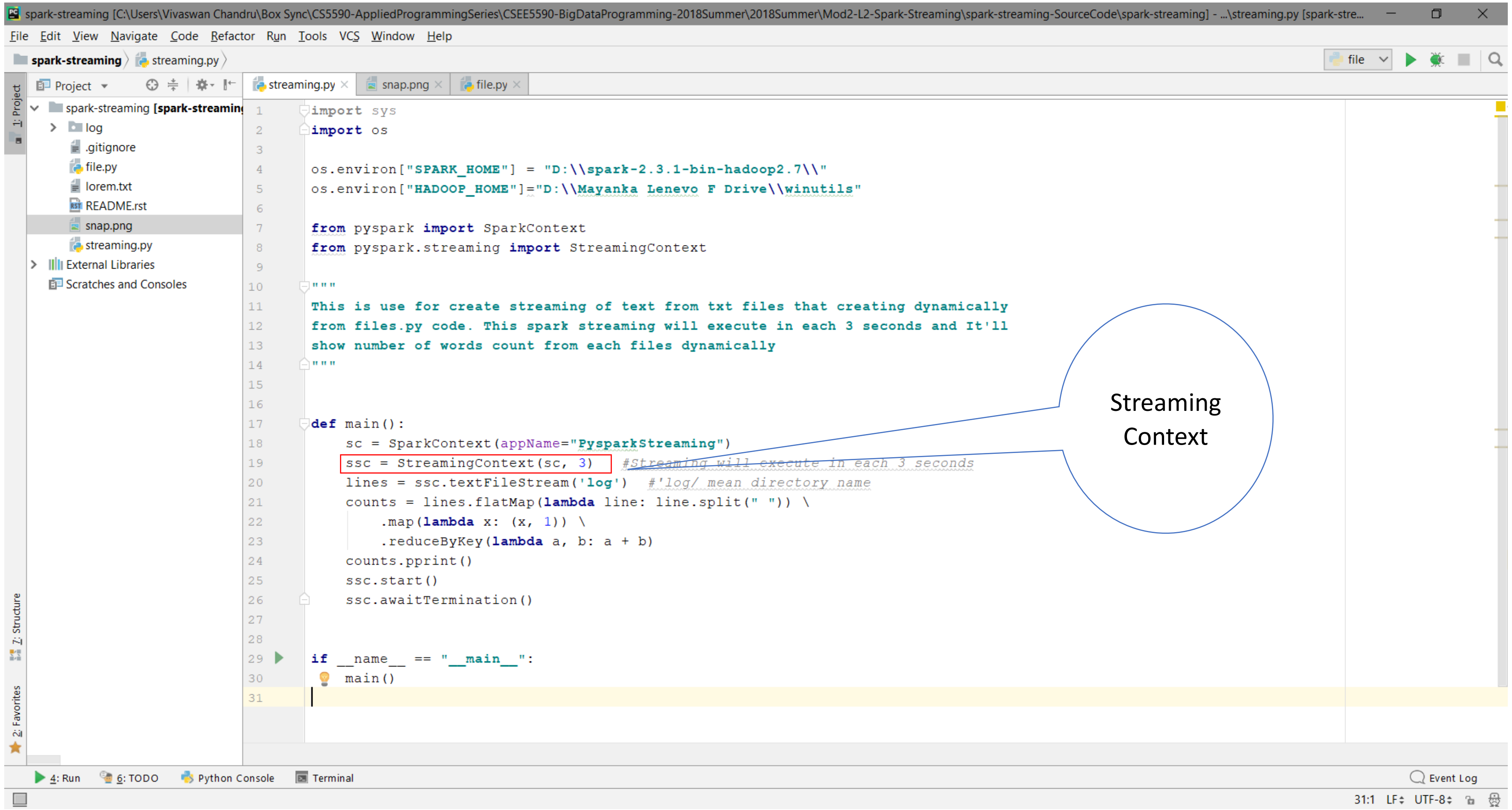
Checkpoints

Data Checkpoints

It is saving of the generated RDDs to reliable storage

Figure: *Features of Checkpoints*

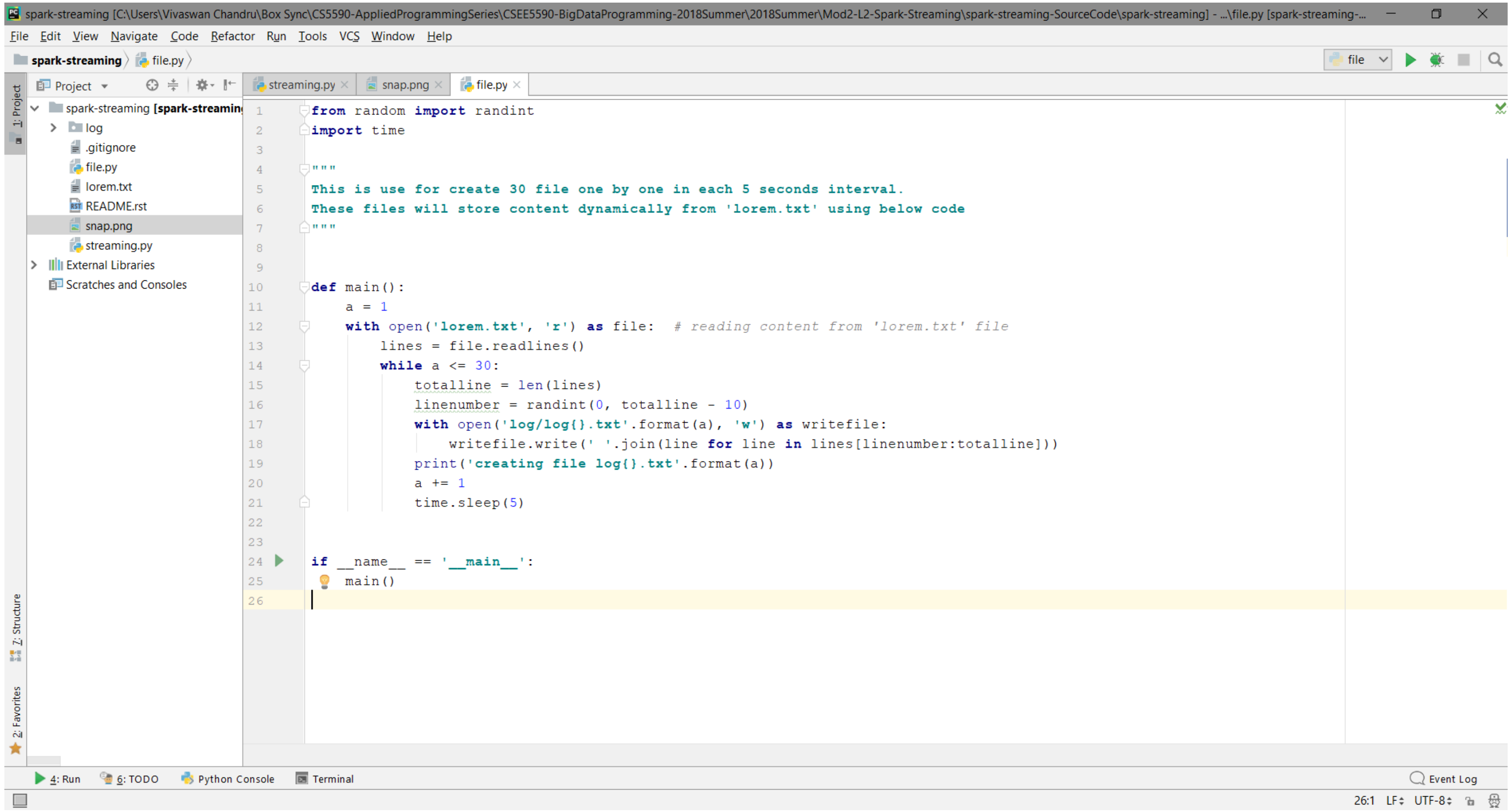
Spark Streaming


spark-streaming [C:\Users\Vivaswan Chandru\Box Sync\CS5590-AppliedProgrammingSeries\CSEE5590-BigDataProgramming-2018Summer\2018Summer\Mod2-L2-Spark-Streaming\spark-streaming-SourceCode\spark-streaming] - ...\streaming.py [spark-stre...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
spark-streaming
streaming.py
file
Project
streaming.py
snap.png
file.py
spark-streaming [spark-streamin
log
.gitignore
file.py
lorem.txt
README.rst
snap.png
streaming.py
External Libraries
Scratches and Consoles
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Python Console
Terminal
Event Log
31:1 LF: UTF-8:

```
import sys
import os

os.environ["SPARK_HOME"] = "D:\\spark-2.3.1-bin-hadoop2.7\\"
os.environ["HADOOP_HOME"]="D:\\Mayanka Lenevo F Drive\\winutils"

from pyspark import SparkContext
from pyspark.streaming import StreamingContext

"""
This is use for create streaming of text from txt files that creating dynamically
from files.py code. This spark streaming will execute in each 3 seconds and It'll
show number of words count from each files dynamically
"""


def main():
    sc = SparkContext(appName="PysparkStreaming")
    ssc = StreamingContext(sc, 3)  #Streaming will execute in each 3 seconds
    lines = ssc.textFileStream('log')  #'log/ mean directory name
    counts = lines.flatMap(lambda line: line.split(" ")) \
        .map(lambda x: (x, 1)) \
        .reduceByKey(lambda a, b: a + b)
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()


if __name__ == "__main__":
    main()

```

Streaming Context

File Generation



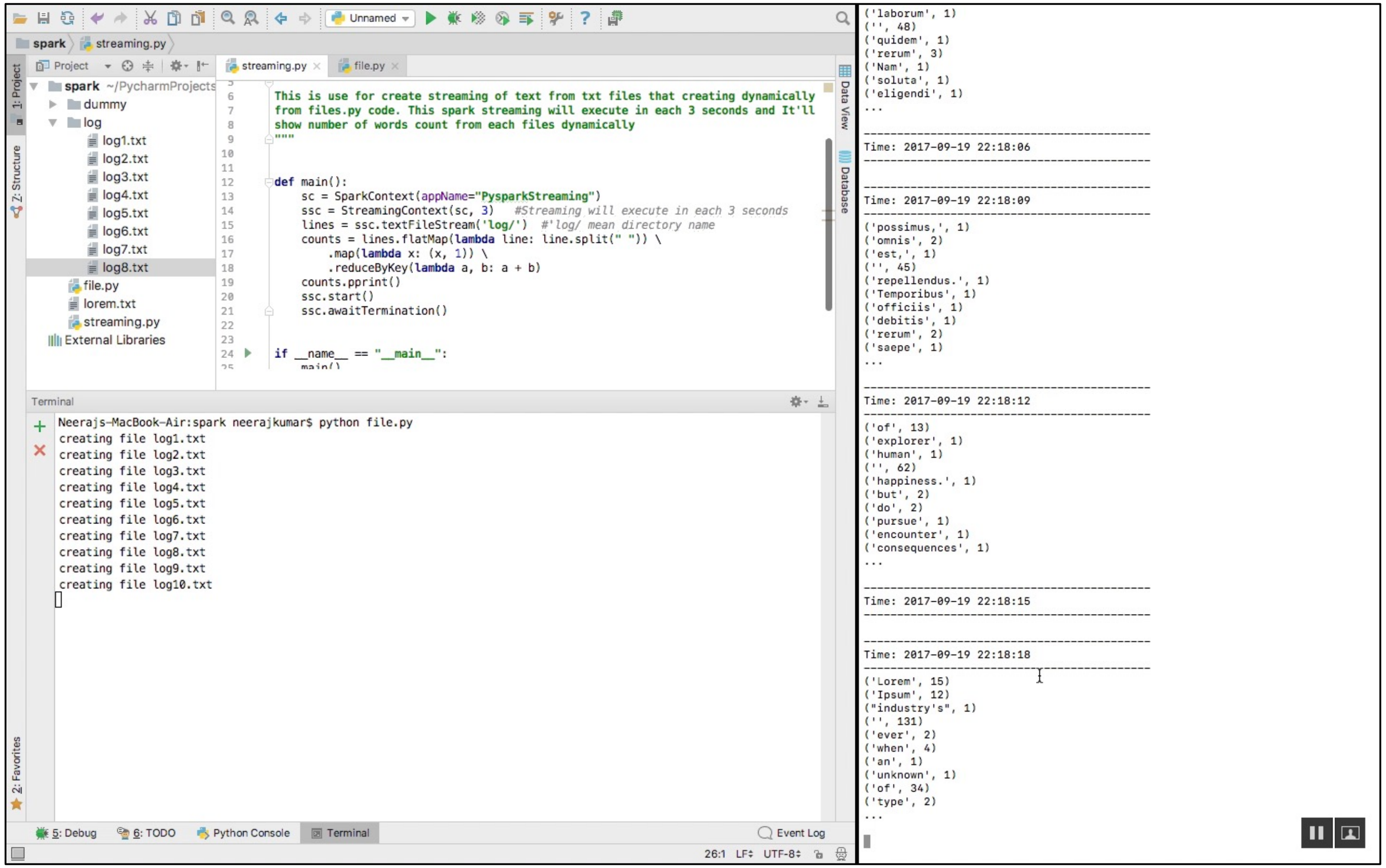
spark-streaming [C:\Users\Vivaswan Chandru\Box Sync\CS5590-AppliedProgrammingSeries\CSEE5590-BigDataProgramming-2018Summer\2018Summer\Mod2-L2-Spark-Streaming\spark-streaming-SourceCode\spark-streaming] - ...\file.py [spark-streaming-...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
spark-streaming
file.py
file
Project
streaming.py
snap.png
file.py
spark-streaming [spark-streamin
log
.gitignore
file.py
lorem.txt
README.rst
snap.png
streaming.py
External Libraries
Scratches and Consoles
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Python Console
Terminal
Event Log
26:1 LF: UTF-8:

```python
from random import randint
import time

"""
This is use for create 30 file one by one in each 5 seconds interval.
These files will store content dynamically from 'lorem.txt' using below code
"""


def main():
    a = 1
    with open('lorem.txt', 'r') as file:  # reading content from 'lorem.txt' file
        lines = file.readlines()
        while a <= 30:
            totalline = len(lines)
            linenumber = randint(0, totalline - 10)
            with open('log/log{}.txt'.format(a), 'w') as writefile:
                writefile.write(' '.join(line for line in lines[linenumber:totalline]))
            print('creating file log{}.txt'.format(a))
            a += 1
            time.sleep(5)


if __name__ == '__main__':
    main()

```

Download Netcat

- Go https://joncraton.org/files/nc111nt.zip
- Unzip the files and use nc as password
- Go to environment variables
- Select Path and Add new path as you location to netcat folder.
- Open command line
- Type nc to check either netcat is working or not

Running Results



```python
This is use for create streaming of text from txt files that creating dynamically
from files.py code. This spark streaming will execute in each 3 seconds and It'll
show number of words count from each files dynamically
"""


def main():
    sc = SparkContext(appName="PysparkStreaming")
    ssc = StreamingContext(sc, 3)   #Streaming will execute in each 3 seconds
    lines = ssc.textFileStream('log/')  #'log/ mean directory name
    counts = lines.flatMap(lambda line: line.split(" ")) \
        .map(lambda x: (x, 1)) \
        .reduceByKey(lambda a, b: a + b)
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()


if __name__ == "__main__":
    main()
```

```
Neerajs-MacBook-Air:spark neerajkumar$ python file.py
creating file log1.txt
creating file log2.txt
creating file log3.txt
creating file log4.txt
creating file log5.txt
creating file log6.txt
creating file log7.txt
creating file log8.txt
creating file log9.txt
creating file log10.txt
```

```
('laborum', 1)
('', 48)
('quidem', 1)
('rerum', 3)
('Nam', 1)
('soluta', 1)
('eligendi', 1)
...

-------------------------------------------
Time: 2017-09-19 22:18:06
-------------------------------------------

-------------------------------------------
Time: 2017-09-19 22:18:09
-------------------------------------------
('possimus,', 1)
('omnis', 2)
('est,', 1)
('', 45)
('repellendus.', 1)
('Temporibus', 1)
('officiis', 1)
('debitis', 1)
('rerum', 2)
('saepe', 1)
...

-------------------------------------------
Time: 2017-09-19 22:18:12
-------------------------------------------
('of', 13)
('explorer', 1)
('human', 1)
('', 62)
('happiness.', 1)
('but', 2)
('do', 2)
('pursue', 1)
('encounter', 1)
('consequences', 1)
...

-------------------------------------------
Time: 2017-09-19 22:18:15
-------------------------------------------

-------------------------------------------
Time: 2017-09-19 22:18:18
-------------------------------------------
('Lorem', 15)
('Ipsum', 12)
("industry's", 1)
('', 131)
('ever', 2)
('when', 4)
('an', 1)
('unknown', 1)
('of', 34)
('type', 2)
...
```

References

- https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html
- https://www.edureka.co/blog/spark-streaming/