CS 199 ACC
Spark

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim
Bhuvan Venkatesh

MP 3

How was it?

This Week

- Distributed Computation
- Apache Spark
- Use Cases

Spark - Intro

Computation Framework Code

- Most of you have not dealt with computational framework code
- This code is different because instead of doing what you want to do you usually tell a class or an object what you want to do and that class translates that instruction into something the engine can understand
- You are building a **pipeline** for your data
  - Problems arise when you have to fix something in your pipeline!
- Other computation frameworks:
  - TensorFlow, Spark, and Cafe

Apache Spark?

- Yay, more Apache!
- Distributed computing framework
    - Basically, you give it code and it'll run on the cluster
- Can be run on top of Hadoop (more details later)
- Fast, Scalable, Optimized, Works on Heterogenous Clusters

Spark Speciality - Resilient Distributed Dataset

- Recall from last week the problems about distributed computation
  - Nodes can go down
  - Network can be unreliable
- What happens if you have a constantly changing dataset and a node goes down? How do you guarantee that the dataset is recovered?
- Enter: Resilient Distributed Dataset (**RDD**) RDDs are immutable data sets, meaning they cannot be changed.
- Let’s think about the cases: If no nodes go down, the RDD gets processed

RDD - Cont

- If a node goes down:
  - If it is the master, a new master is elected and then proceeds below
  - If not, the master determines what part of the RDD that the node was working on. Since RDDs cannot be changed, another node has a copy of that part. That node then completes the computation
- If there is a network partition:
  - The cluster always picks the side that the master is on. The master then figures out what parts of the RDD need to be recomputed
- Because RDDs are Immutable: we have an inherent “checkpoint” so if a node goes down then we can back the computation up and rerun

Basic Operations

Okay So What Does Some Code Look Like?

```python
count = len([line for line in \
    open('file.txt') \
    if 'pattern' in line])
print(count)
```

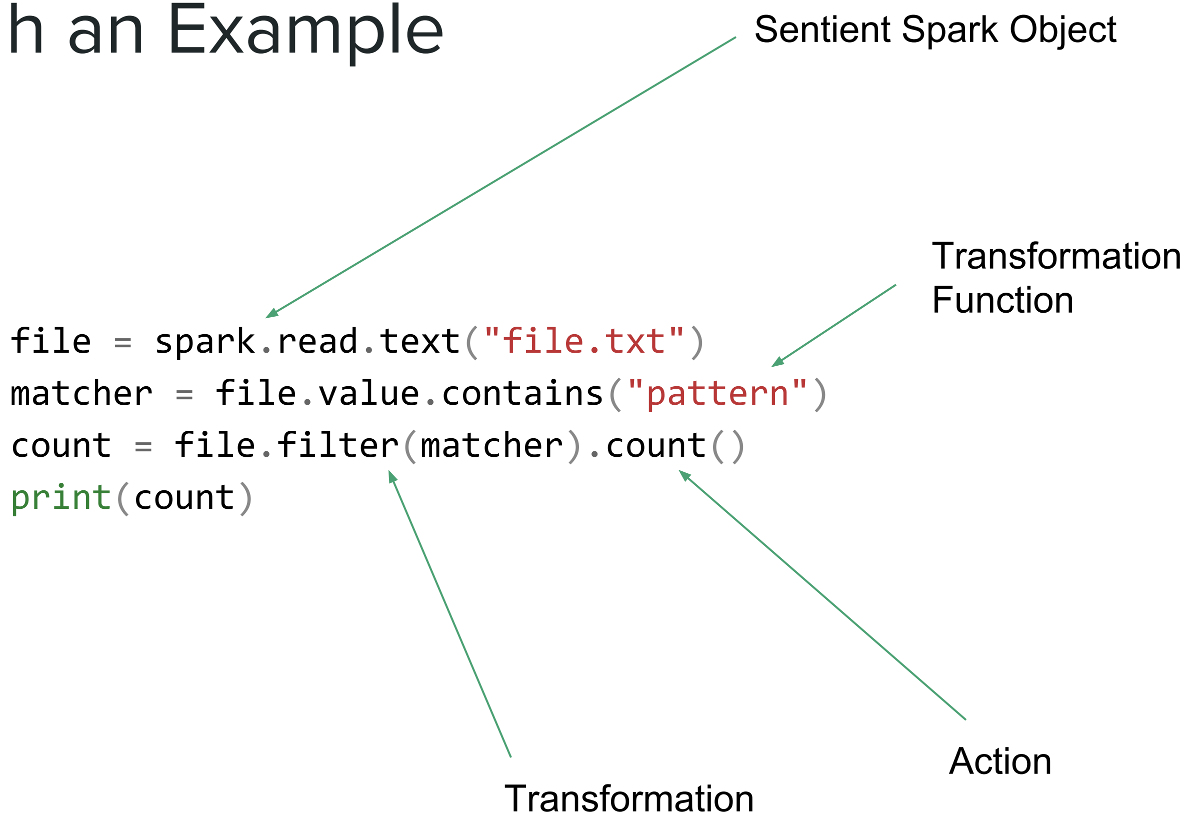
```python
file = spark.read.text("file.txt")
matcher = file.value.contains("pattern")
count = file.filter(matcher).count()
print(count)
```

RDD Transformation

- How do you alter RDDs? You apply a transformation. There are 2 types:

- **Transformations**: These are things like Map, Filter. They take a row and either turn it into a different row or remove the row entirely.
  - Transformations are lazy, meaning that they are not evaluated right away
- **Actions**: These are things like Reduce, Count. They usually produce some kind of “result”, whether that be something that you are going to use in the next computation or something you are going to print to a file.
  - This produces a new RDD

Going Through an Example



```
file = spark.read.text("file.txt")
matcher = file.value.contains("pattern")
count = file.filter(matcher).count()
print(count)
```

Nitty Gritty

Memory

- Spark ~~devours~~ likes memory
- It tries to do as much of its computation sequentially and in memory
- Spark tries to cache as much data as possible
- Natural Consequence: you need to give Spark a lot of real memory to prevent inefficiency

Execution

- You’ve given spark an action, what does it do then?
- A leader that knows where each of the relevant pieces of data are takes the query and divides it up into “dependencies”.
- This forms a directed acyclic graph. If a node fails, one just has to do a reverse search to find the piece of data that needs to be computed.

Spark Optimizer

- Spark optimizes the code based on squashing map and filters
- Spark localizes data so that one machine will try to do most of the computation
- That node will check back in with the leader at stages
- Spark will avoid replicating intermediate data frames
- This is why we need something like a scripting analysis so spark can take a look at the source at a higher level

Data Frames

- Alternatives to RDDs
- Not resilient
- But waaaaaaay faster
- You can edit in line, but no new data frames are created
- You can't use them for most MLlib algorithms as of yet

Spark Modules

Spark Streaming

- Process data as a stream
- Kinda clunky. What we do is we process multiple RDDs as they come in. This means that we don’t have all the data, so we write the algorithms with that knowledge.
- Identify your use case
  - Are you writing an algorithm for some kind of feedback loop?
  - Are you writing a program that displays a running summary?
  - Are you writing for high throughput?
- Really designed for batch jobs

Spark SQL

- You can run SQL queries on datasets
- SQL optimizes things like JOINS
- Since everything is in memory, can induce a lot of thrashing
- There may not be a concept of an “index” so every operation may be O(n). If the data is very split up, joins could flood the network with traffic.
- We will teach it to you but ideally there should be a platform optimized for running SQL

Spark MLIB

- The most used spark extension
- You can feed a dataset to an algorithm and either train or predict on it
- You can grab the internals of a model and use it to describe business logic
- It is pretty hard to tweak a model in a small way -- a lot of people opt for building their own models because of this
- You have to use RDDs which are ehhhhh (Spark also supports dataframes which are generally easier to work with)

GraphX

- Please don’t
- Just kidding, but spark has some definite drawbacks with graphs
- Since graphs are essentially random access elements, it takes away improvements like caching
- A **lot** of network traffic may be generated because the node doesn’t “belong” to the current computer
- Pregel is hard to use for the first time around

MP 4

Due in one week (10/4) at 11:55pm

Warning!

- We **don’t guarantee** the cluster uptime
- Even though Hadoop is scalable, reliable, and fault tolerant (all those buzzwords), imagine what would happen if all thirty of you tried to log on to the cluster and submit a huge mapreduce job at the deadline
  - Either the cluster crashes or it runs at a snail’s pace.
- As with course policy, if it’s late it is late.