CS199: Applied Cloud Computing

Intro to Spark

Prof. Robert J. Brunner

Quinn Jarrell
Tyler Kim

Lab 3

How was it?

Intro to Spark

- Spark is a popular new approach to processing Big Data.
- Spark extends the **MapReduce** model to support more types of computations using a ***functional programming paradigm***
    - You will learn more about functional programming in CS421
- It can cover a variety of workflows that were previously implemented thru special systems built on top of Hadoop
    - Integration with Hadoop
        - No separate storage layer
        - Excellent integration and interaction with other ecosystems.

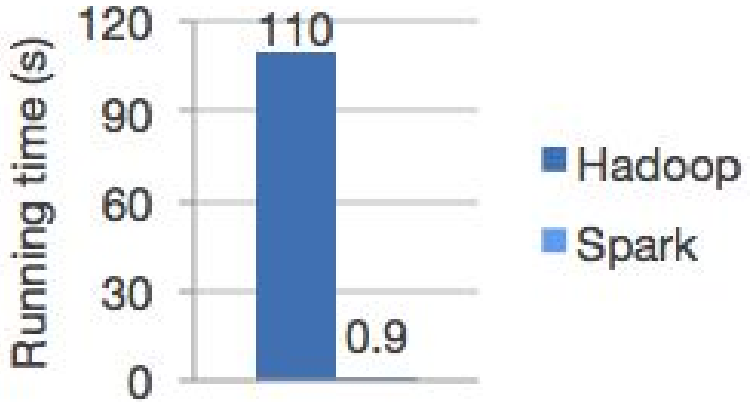
So Why Spark?

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory.

Spark can **cache** HDFS data in main memory of each nodes. Analysis can be executed directly on in memory data

This week's lab will run faster! :D



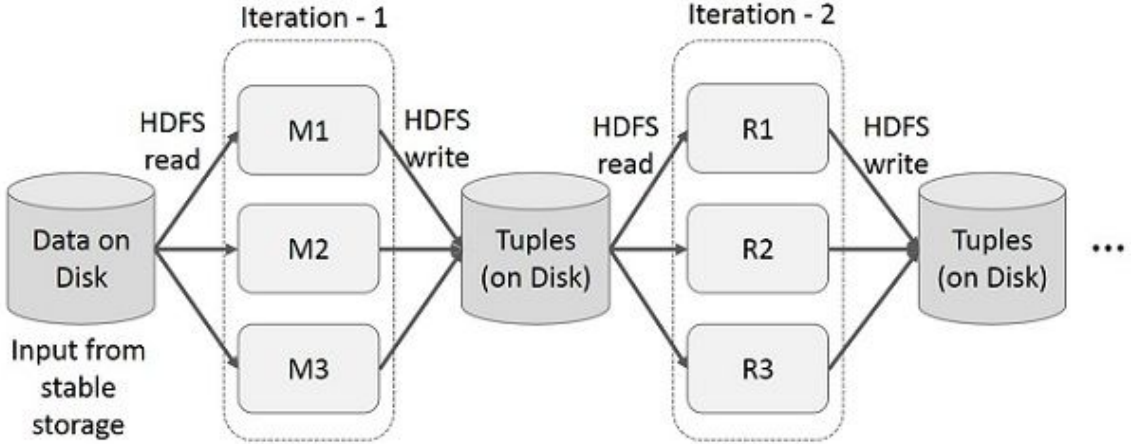
Logistic regression in Hadoop and Spark

How is it so fast?

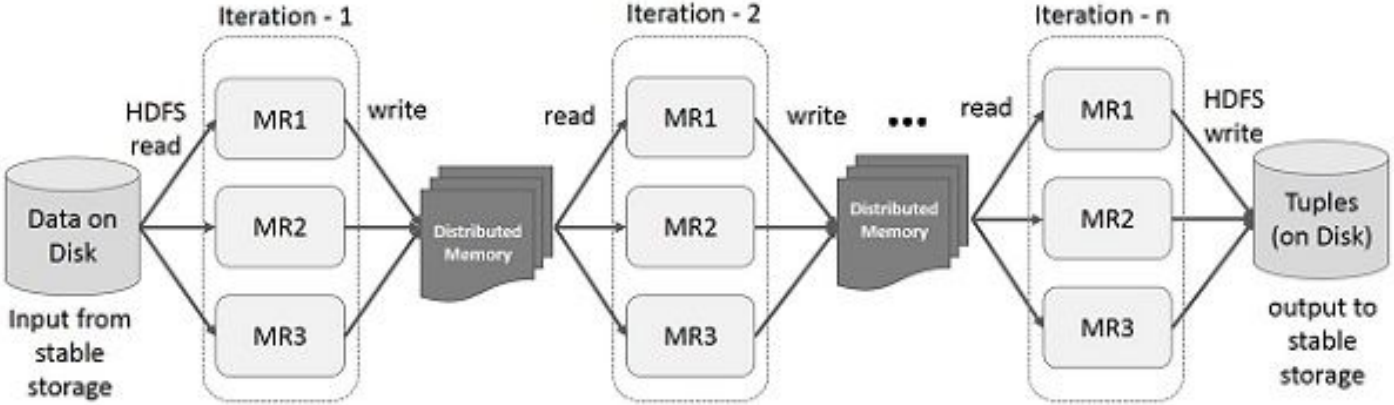
- A little bit of history
  - Spark was developed at the AMPLab at UC Berkeley
  - It took all the bad design decisions in Hadoop and fixed them
    - Hindsight is 20/20
  - Turns out RAM decreased in price a lot in the past few years
- When Hadoop processes a map or reduce stage it must write the output to disk
  - Writing to disk is SLOW compared accessing memory
  - A disk write is measured in milliseconds, main memory writes take nanoseconds!!
  - Spark tries to keep as much in memory as possible

Spark is speedy

Hadoop



Spark



So why have you been using Hadoop?

- Spark is almost equivalent to Hadoop
  - Why did we teach you hadoop?

So why have you been using Hadoop?

- Spark is almost equivalent to Hadoop
  - Why did we teach you hadoop?
- **Almost** is not exactly equivalent
  - Hadoop is more battle tested and has scaled to crazy workloads
  - Spark is getting there and we should be able to use it for the rest of class
  - Hadoop forces you to understand MapReduce
- We have a ton more disk space than RAM right now

You are going to love Spark

- In Spark you write plain old python without having to consider reading STDIN and outputting the right output to STDOUT
- This is valid spark code

```
def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample).reduce(lambda a, b: a + b)
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

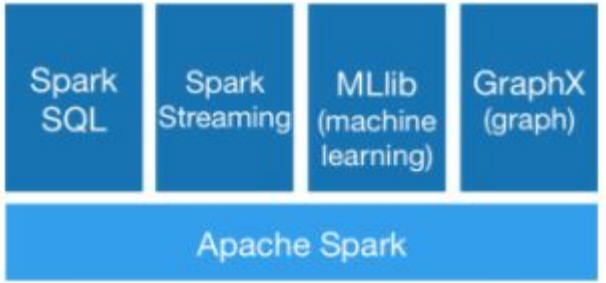
Spark Ecosystem

Unified Platform for Big Data

- It runs on everywhere!
  - Hadoop (1.x & 2.x)
  - Mesos
  - NoSQL
  - GraphX
  - Or on its own!

Generality

- Stack Library
  - SQL, GraphX, MLlib, Streaming, NoSQL, etc.



RDD - Resilient Distributed Dataset

Spark is built on a concept called Resilient Distributed Datasets (RDD).

- Each dataset is split up amongst the computers
  - Sort of like MR sort but much more fine-grained
- Spark tries to store as much as possible in memory not in disk
- RDDs are read only. When you process an RDD you get a new RDD back, it does not change the old one
- Spark is lazy. It will not evaluate an operation until it is absolutely needed

RDD - Resilient Distributed Dataset

- RDDs are made up of partitions
  - A partition is a section of data
  - For instance a date could be a partition
  - In general you want your partitions to contain many elements but not all
  - Each partition is handled by a separate node
  - Do not have one partition
    - One node will try to run it
  - Do not have number of partitions == len(data)
    - It will start your python code in a new process for EVERY element
      - This is really really expensive and will slow you down

How lazy is Spark?

- Spark has two main types of operations
  - Transformations
    - Stuff like map, reduce, filter that return a new dataset but do not run immediately
  - Actions
    - Operations that return answers and force Spark to evaluate your transformations
- I can write a bunch of transformations but Spark will not actually run anything until I run an action

New Transformations

- You already know map and reduce
- There’s more!
- Filter
  - Loops over an array and discards items based on your function

Nums = [0,1,2,3,4,5,6,7]

print nums.filter(lambda x: x % 2 == 0)

Nums == [0,2,4,6]

New Transformations

- You already know map and reduce
- There’s more!
- Filter
  - Loops over an array and discards items based on your function

Nums = [0,1,2,3,4,5,6,7]

print nums.filter(lambda x: x % 2 == 0)

Nums == [0,2,4,6]

Demos!

Tips

- Repartition increases parallelism
- Reduce is really really slow so avoid it

Lab 4

Lab 4 will come out on Friday night, not tonight.

Lab 4 will be due in 2 weeks next Friday (March 3rd) at 11:55pm