Lecture 6: More Spark

Professor Robert J Brunner
Quinn Jarrell

Lab 4

How was it?

Lab 4

How was it?

- Do it early
- Cluster has been upgraded a bit
- It will still slow down next Thursday night
- Ask questions early

Spark Jargon

- Spark has a lot of jargon for things which are very similar in function but work at different scale
- Like managers of managers of managers
  - Each does the same job for a separate level

Tasks

- A task is your function running on a particular partition
- In general if you have X partitions, you have X tasks *per transformation*
- Think of it as a thread

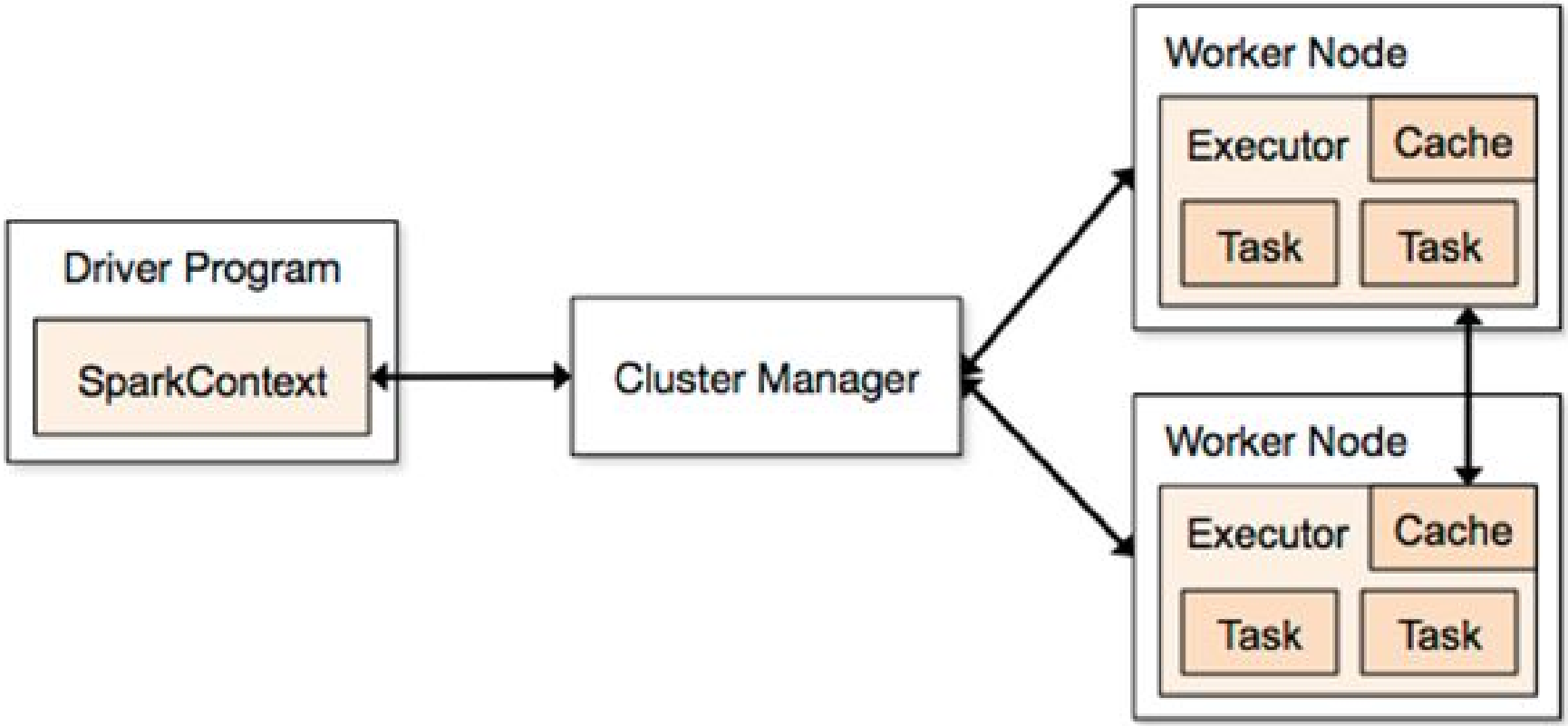
Executors

- Acts as a supervisor for the tasks
- Reports progress to driver
- Multiple per computer
  - Why?

Driver

- Only one of it per job
- Acts as a coordinator for executors
- If using PySpark binary, it is the driver
- If using spark-submit, creates a driver either locally or elsewhere on the cluster

Putting it all together



VCores

- Each computer has a number of virtual CPUs.
- Most nodes in our cluster have 8

Guidelines on optimization

- Spark by default does not allocate more than a minimum of executors
- When running on the whole test set, you should set the number of executors and VCores higher
  - Don’t set it too high or we will kill your job
  - We’ll let you know how high to set it, it depends on how busy the cluster is
- You want more executors than VCores
  - Why?
- Check the number of partitions
  - If it’s too few, repartition

Compiler Ambiguity

- Why can Spark auto-parallelize stuff but your normal compiler like GCC or LLVM can’t?

Compiler Ambiguity

- Why can Spark auto-parallelize stuff but your normal compiler like GCC or LLVM can’t?
- Well for a normal mutable language, you cannot predict what parallelization would work best
- This ambiguity of what a loop is going to do stops GCC or LLVM being smarter
    - They can auto-parallelize patterns they recognize but not everything

Relaxing Constraints

So since we can’t solve the Halting Problem, what can we do?

Relaxing Constraints

So since we can’t solve the Halting Problem, what can we do?

- Alter the language itself
- If everything is immutable we can parallelize without fear of side effects
- Force us to use easily parallelizable patterns

Immutability Round Two

- Remember how immutability works?
- No side effects
- f(x,y) == f(x,y) ALWAYS

For loops vs Map

- We know a for loop is almost always equivalent to a map operation

```
Arr = [0,1,2,3]

For i in arr:

    new_arr.append(i**2)
```

```
Arr = [0,1,2,3]

New_arr = map(lambda x: x**2, arr)
```

For loops vs Map

- We know a for loop is almost always equivalent to a map operation

```
Arr = [0,1,2,3]

For i in arr:

    new_arr.append(i**2)
```

```
Arr = [0,1,2,3]

New_arr = map(lambda x: x**2, arr)
```

- So why does Spark require us to use map/reduce/filter?

For loops vs Map

- Well for loops are mutable

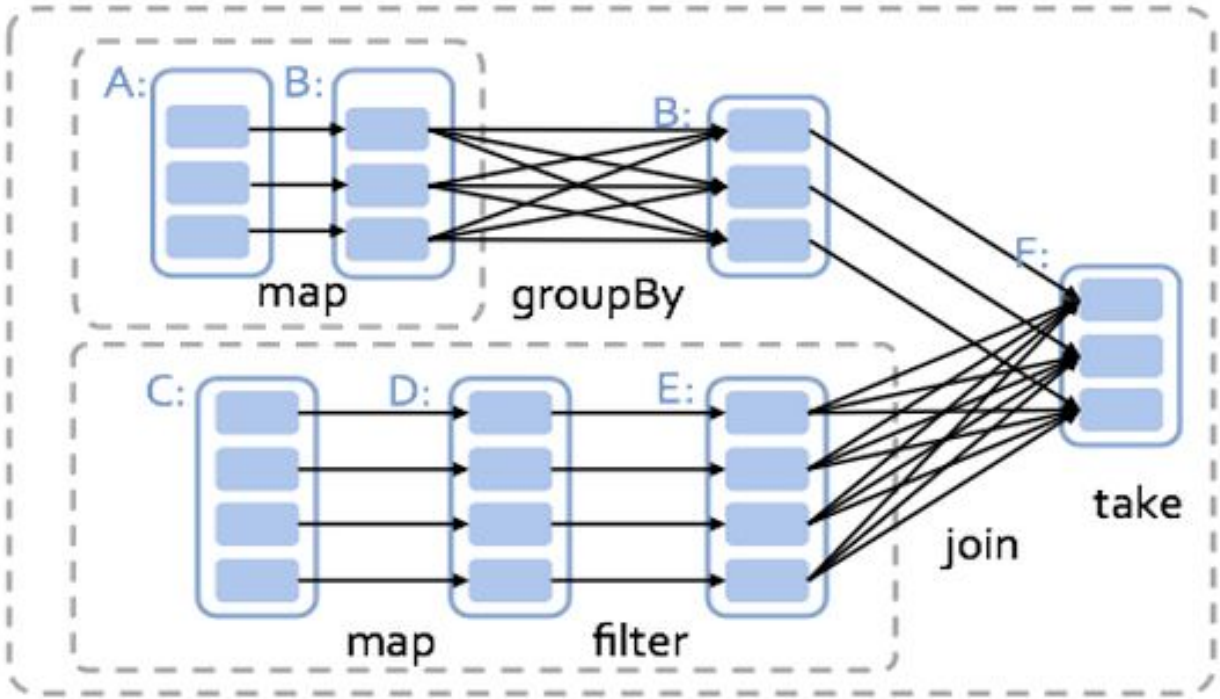
```
Arr = [0,1,2,3]

For i in arr:

    Arr[i] = (i**2)
```

- This modifies Arr

Spark Compiler

- Spark creates a graph of operations.

- It divides operations into stages
- Each stage ends when a reduce/shuffle happens



Demo!

Spark MLib

- Lets you do distributed machine learning on massive datasets
- Built on top of Spark so you can use it without switching to a new system
- Designed to look like NumPy
- Can interoperate with NumPy and SciPy

Spark Datatypes

MLib deals with two main datatypes

- Dense Vectors
  - Dense is what you are used to
  - Vector = [1.0,0.0,3.0]
- Sparse Vectors are new
  - They include indices of none zero elements only
  - sv1 = Vectors.sparse(3, [0, 2], [1.0, 3.0])
  - Sv1 has 3 elements total, with none zero elements at indices 0 and 2
  - Actually looks like
    - [1.0,0.0,3.0]

Sparse Vectors

- Use sparse vectors when a lot of elements can 0
- They compress better and Spark has customized functions for dealing with it faster
- Sparsity is common
    - While raw data is not normally sparse, the moment you start filtering it becomes sparser and sparser

Matrixes

- Matrices are common in machine learning
- Spark has normal local matrices where each matrix resides entirely on one computer
- But what happens when you have 100 GB matrices

Partitioning Matrices

- Most of the time you will have to set the partitioning type for matrices
- By using Spark, your data better be big
- You need to partition a matrix so that operations do not have to shuffle partitions around as little as possible
- RowMatrices have each row be a partition
  - Great for row operations, terrible for column operations (Hint, transpose)

  [[part1],[part2],[part3]]

  They support add and multiply by a local matrix (Not distributed)

- BlockMatrices partitions the matrix by smaller rectangular matrixes

Row Matrices

- RowMatrices have each row be a partition
  - Great for row operations, terrible for column operations (Hint, transpose)

  [
  [part1],
  [part2],
  [part3]
  ]
  They support add and multiply by a local matrix (Not distributed)

- Until you get to massive massive matrix multiplies, you'll probably only use row matrices

Block Matrices

- Matrix is split into blocks of submatrices. Each block is a partition
- Only supports multiplying by another block matrix
- This should have no size limit but will be much slower than row matrices when smaller

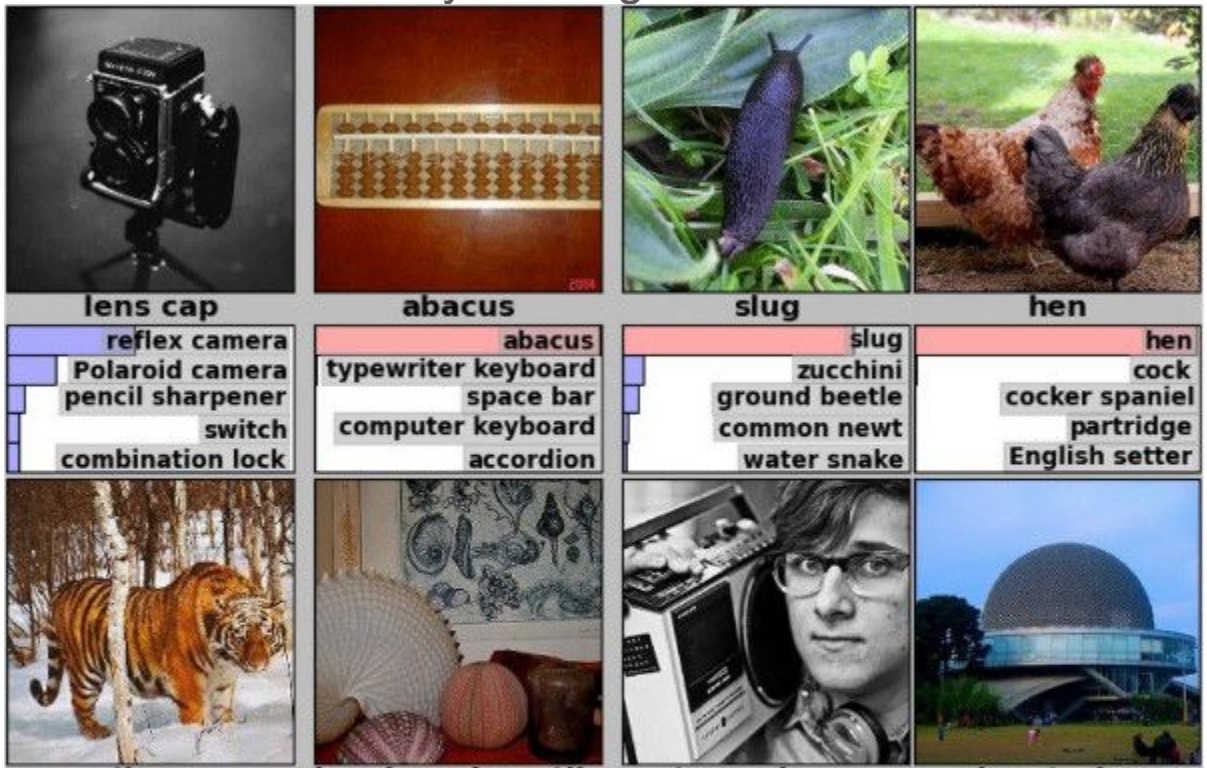
Data Locality

- You've probably noticed a pattern by now
  - The network is slow

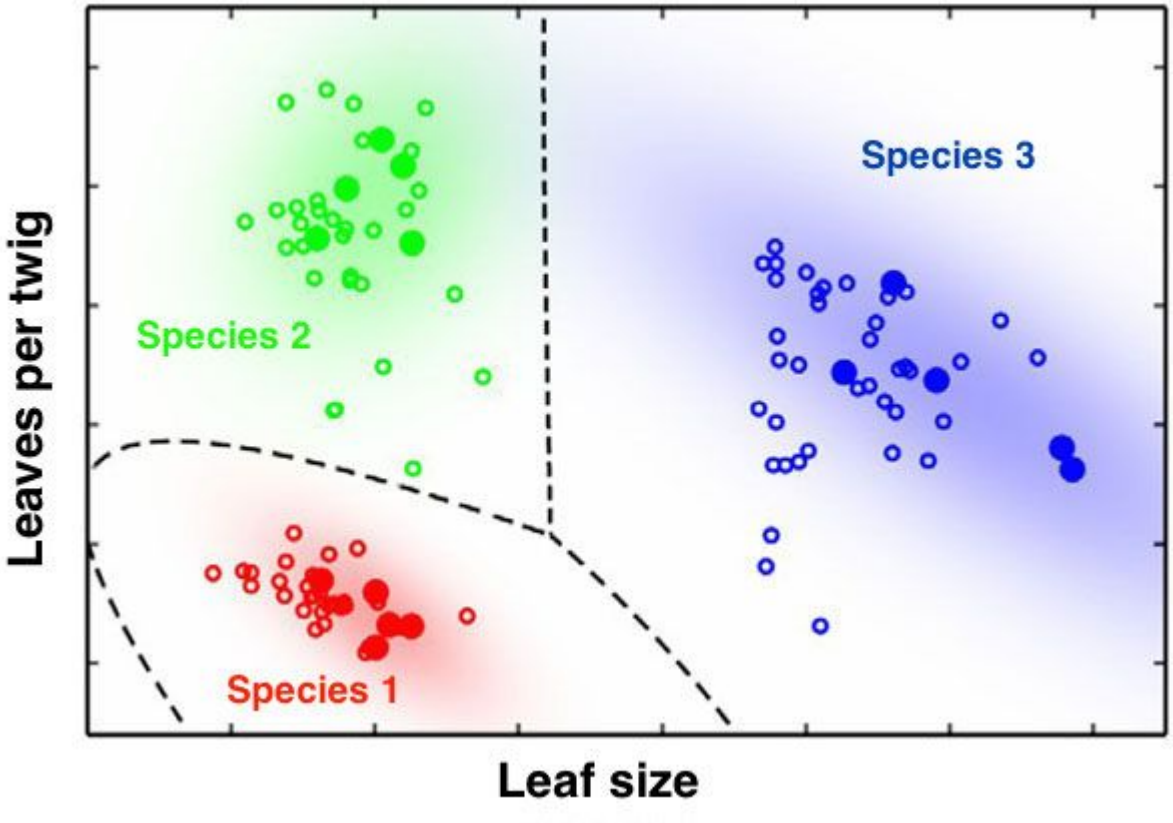
Main memory reference	100 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Classifiers

- Classifiers take in data and try to assign a label or labels to it

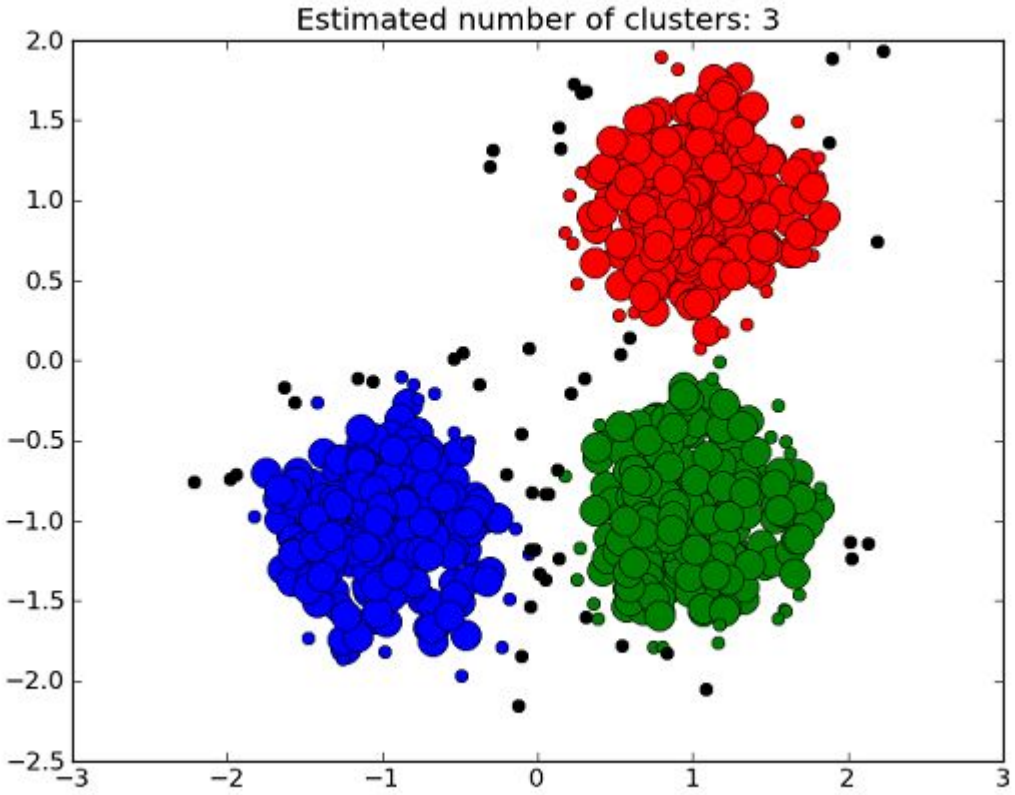


Classifiers



Clustering

- Don’t know the number or what the labels are



When you should use MLib

- Obviously when you have extremely large data
- When it is not GPU intensive
  - A lot of machine learning benefits from a single GPU than 100 CPUs

Lab 4 Advice

- Again, don’t wait until Thursday
- There are really good programming guides online for spark

Projects

Start thinking of ideas

Start building groups

Start identifying what technologies you want to use

- We’ll teach you new ones if we haven’t covered the one you want