L06: Introducing Spark

ANLY 502: Massive Data Fundamentals

Simson Garfinkel & Ghaleb Abdulla

February 28, 2016

Outline for today's class

PS03 Redux

Student Presentations

Spark's architecture.

RDDs

Writing Spark programs in Scala and Python

Internet sources for information about Spark.

Word Count in Spark.

What's in Web Logs.

Log file analysis with Spark.

But first, three words...

It's getting easier!

We've done a lot in the past six weeks!

What we've done:

- Unix command line
- Virtualization with VMWare
- Cloudera VM
- Amazon Web Services
- Spending real money — $$$
- MapReduce
- mrjob
- Pig
- Debugging
- Gigabyte-sized data sets
- 5 minute presentations

We've struggled with:

- Buggy software
- Inconsistent behavior and error.
- Insufficient documentation.
- Differences between runtime environments.

Where we're going:

- Spark
- SparkSQL
- Data wrangling
- Terabyte-sized data sets
- LLNL Lectures
- Class Projects

We've done a lot in the past six weeks!

What we've done:

- Unix command line
- Virtualization with VMWare
- Cloudera VM
- Amazon Web Services
- Spending real money — $$$
- MapReduce
- mrjob
- Pig
- Debugging
- Gigabyte-sized data sets
- 5 minute presentations

We've struggled with:

- Buggy software
- Inconsistent behavior and error.
- Insufficient documentation.
- Differences between runtime environments.

Where we're going:

- Spark
- SparkSQL
- Data wrangling
- Terabyte-sized data sets
- LLNL Lectures
- Class Projects

https://pixabay.com/en/student-typing-keyboard-text-woman-849825/

PS03 Redux

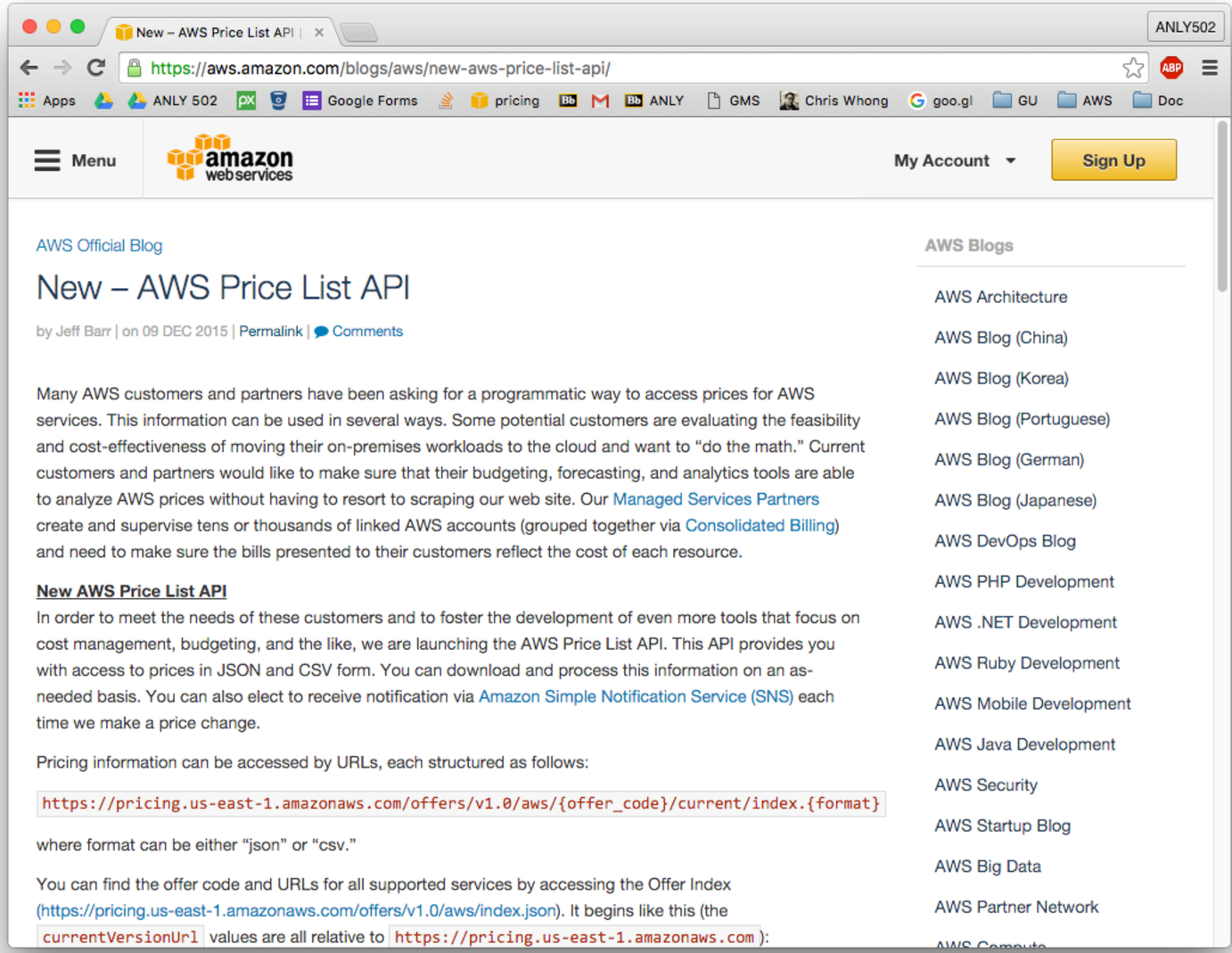
Using Amazon's API to find out about real-time pricing

AWS Price List API

- Offers — Services AWS is offering
- Products — e.g. VM instances
- Terms — e.g. OnDemand, Annual, etc.

Example:


```
"RDXNGJU5DRW4G5ZK" : {
  "sku" : "RDXNGJU5DRW4G5ZK",
  "productFamily" : "Compute Instance",
  "attributes" : {
    "servicecode" : "AmazonEC2",
    "location" : "South America (Sao Paulo)",
    "locationType" : "AWS Region",
    "instanceType" : "c3.large",
    "currentGeneration" : "Yes",
    "instanceFamily" : "Compute optimized",
    "vcpu" : "2",
    "physicalProcessor" : "Intel Xeon E5-2680 v2 (Ivy Bridge)",
    "clockSpeed" : "2.8 GHz",
    "memory" : "3.75 GiB",
    "storage" : "2 x 16 SSD",
    "networkPerformance" : "Moderate",
    "processorArchitecture" : "32-bit or 64-bit",
    "tenancy" : "Host",
    "operatingSystem" : "Linux",
    "licenseModel" : "No License required",
    "usagetype" : "SAE1-HostBoxUsage:c3.large",
    "operation" : "RunInstances",
    "enhancedNetworkingSupported" : "Yes",
    "preInstalledSw" : "NA",
    "processorFeatures" : "Intel AVX; Intel Turbo"
  }
},
```

```
#!/usr/bin/env python3.5
offer_index_url = "https://pricing.us-east-1.amazonaws.com/offers/v1.0/aws/index.json"
import json,urllib.request
from tabulate import tabulate

if __name__=="__main__":
    offers = json.loads(urllib.request.urlopen(offer_index_url).read().decode('utf-8'))
    assert(offers['formatVersion']=='v1.0')

    table = [["Offer","offerCode","currentVersionUrl"]]

    for name in offers['offers']:
        od = offers['offers'][name]
        table.append([name,od['offerCode'],od['currentVersionUrl']])
    print("The following offers are available:")
    print(tabulate(table,headers="firstrow",tablefmt="simple"))
```

Print the offers!

- Output:

```
$ ./aws_costing.py
The following offers are available:
Offer              offerCode          currentVersionUrl
-----------------  -----------------  ------------------------------------------------------
AmazonRedshift     AmazonRedshift     /offers/v1.0/aws/AmazonRedshift/current/index.json
AmazonSimpleDB     AmazonSimpleDB     /offers/v1.0/aws/AmazonSimpleDB/current/index.json
AmazonRDS          AmazonRDS          /offers/v1.0/aws/AmazonRDS/current/index.json
AmazonSES          AmazonSES          /offers/v1.0/aws/AmazonSES/current/index.json
AmazonRoute53      AmazonRoute53      /offers/v1.0/aws/AmazonRoute53/current/index.json
AmazonVPC          AmazonVPC          /offers/v1.0/aws/AmazonVPC/current/index.json
awskms             awskms             /offers/v1.0/aws/awskms/current/index.json
AmazonEC2          AmazonEC2          /offers/v1.0/aws/AmazonEC2/current/index.json
AmazonElastiCache  AmazonElastiCache  /offers/v1.0/aws/AmazonElastiCache/current/index.json
AmazonS3           AmazonS3           /offers/v1.0/aws/AmazonS3/current/index.json
AmazonCloudFront   AmazonCloudFront   /offers/v1.0/aws/AmazonCloudFront/current/index.json
AmazonDynamoDB     AmazonDynamoDB     /offers/v1.0/aws/AmazonDynamoDB/current/index.json
AmazonGlacier      AmazonGlacier      /offers/v1.0/aws/AmazonGlacier/current/index.json
```

```
# Get the EC2 offers
assert ec2_code in offers['offers']
ec2_url  = base+offers['offers'][ec2_code]['currentVersionUrl']
ec2_json = urllib.request.urlopen(ec2_url).read().decode('utf-8')

# Get all of the current ec2 offers
ec2_products = ec2_info['products']
print("Number of products available: {}".format(len(ec2_products)))
ec2_terms    = ec2_info['terms']
print("Terms available: {}".format(" ".join(ec2_terms)))

# Assemble an array of the instance types

instances = []
for (product,vals) in ec2_products.items():
    try:
           patts = vals['attributes']
           for(pk,pv) in  ec2_terms['OnDemand'][product].items():
                for(dk,dv) in pv['priceDimensions'].items():
                    gb   = float(patts['memory'].replace("GiB",""))
                    vcpu = float(patts['vcpu'])
                    row = (product, # row[0]
                           patts['instanceType'], # row[1]
                           vcpu,         # row[2]
                           gb, # row[3]
                           gb/vcpu,                 # row[4]
                           float(dv['pricePerUnit']['USD'])) # row[5]
                    instances.append(row)
    except KeyError:
        # Missing data
        pass
```

Print the VMs

Convert text to float

9699 different product codes...

product	instanceType	vCPU	Memory	GiB/cpu	pricePerUnit
SZAG69AWYJF676BA	d2.4xlarge	16	122	7.625	0
DSG34N2933CDGRJJ	c4.xlarge	4	7.5	1.875	0
PYCJPPPYA7FXP2KM	m4.large	2	8	4	0
J28DJ6QCZ8VU7DZQ	d2.8xlarge	36	244	6.77778	6.198
62G57MU6KCQ2AQS8	t1.micro	1	0.613	0.613	0.08
7MVN3GT6EP25KDUJ	cc2.8xlarge	32	60.5	1.89062	2
232CDFDW89ENUXRB	d2.8xlarge	36	244	6.77778	0
H3H6PVAND793CJ85	c4.8xlarge	36	60	1.66667	0
86FEVXHAJVJ75D5R	c4.2xlarge	8	15	1.875	0.773
K4FQKJH96JE6DDW2	m3.xlarge	4	15	3.75	0
QZS65ZVZAUNM545N	hi1.4xlarge	16	60.5	3.78125	3.23
XBYJG3BUDTPN8NB9	cc2.8xlarge	32	60.5	1.89062	2.57
TB8JSDKA7MEGTRXV	m4.large	2	8	4	0.132
Q73NFXYCVJRVJD5P	i2.8xlarge	32	244	7.625	9.836
VN8JS6C4CHVEY8WD	i2.4xlarge	16	122	7.625	0.1
ERPWM7KEFVQABEK6	m3.xlarge	4	15	3.75	0
MHX8TSHV6Z45N5KU	d2.xlarge	4	30.5	7.625	0.759
GK3JQYYZHNZAHQ66	r3.2xlarge	8	61	7.625	0
FRR3BPV6Y433HGXY	d2.8xlarge	36	244	6.77778	6.198
PA99ECAE74DADX5J	c4.4xlarge	16	30	1.875	2.408

...

Restrict to N. Virginia, Shared, Linux, and price>0

```
# Get the EC2 offers
assert ec2_code in offers['offers']
ec2_url  = base+offers['offers'][ec2_code]['currentVersionUrl']
ec2_json = urllib.request.urlopen(ec2_url).read().decode('utf-8')

# Get all of the current ec2 offers
ec2_products = ec2_info['products']
print("Number of products available: {}".format(len(ec2_products)))
ec2_terms    = ec2_info['terms']
print("Terms available: {}".format(" ".join(ec2_terms)))

# Assemble an array of the instance types

instances = []
for (product,vals) in ec2_products.items():
    try:
        patts = vals['attributes']
        if patts['location']=='US East (N. Virginia)' and \
                patts['tenancy']=='Shared' and \
                patts['operatingSystem'] == 'Linux':
            for(pk,pv) in  ec2_terms['OnDemand'][product].items():
                for(dk,dv) in pv['priceDimensions'].items():
                    gb   = float(patts['memory'].replace("GiB",""))
                    vcpu = float(patts['vcpu'])
                    row = (product, # row[0]
                           patts['instanceType'], # row[1]
                           vcpu,         # row[2]
                           gb, # row[3]
                           gb/vcpu,                # row[4]
                           float(dv['pricePerUnit']['USD'])) # row[5]
                    # Convert values as necessary
                    if row[4]>0:
                        instances.append(row)
    except KeyError:
        # Missing data
        pass
```

Select N. VA / Shared / Linux

Convert text to float

54 total

product	instanceType	vCPU	Memory	GiB/cpu	pricePerUnit
RJZ63YZJGC58TPTS	hi1.4xlarge	16	60.5	3.78125	3.1
AGHHWVT6KDRBWTWP	t2.nano	1	0.5	0.5	0.0065
3DX9M63484ZSZFJV	cc2.8xlarge	32	60.5	1.89062	2
3UP33R2RXCADSPSX	m4.4xlarge	16	64	4	0.958
VHC3YWSZ6ZFZPJN4	m4.2xlarge	8	32	4	0.479
QY3YSEST3C6FQNQH	t2.medium	2	4	2	0.052
A67CJDV9B3YBP6N6	g2.8xlarge	32	60	1.875	2.6
2GCTBU78G22TGEXZ	m1.small	1	1.7	1.7	0.044
5KHB4S5E8M74C6ES	i2.xlarge	4	30.5	7.625	0.853
ZESHW7CZVERW2BN2	i2.4xlarge	16	122	7.625	3.41
6TEX73KEE94WMEED	c1.xlarge	8	7	0.875	0.52
P63NKZQXED5H7HUK	d2.2xlarge	8	61	7.625	1.38
RKCQDTMY5DZS4JWT	m2.4xlarge	8	68.4	8.55	0.98
X4RWGEB2DKQGCWC2	c1.medium	2	1.7	0.85	0.13
ASDZTDFMC5425T7P	m3.medium	1	3.75	3.75	0.067
QG5G45WKDWDDHTFV	t2.large	2	8	4	0.104
48VURD6MVAZ3M5JX	g2.2xlarge	8	15	1.875	0.65
639ZEB9D49ASFB26	t1.micro	1	0.613	0.613	0.02
U7343ZA6ABZUXFZ9	d2.xlarge	4	30.5	7.625	0.69
NARXYND9H74FTC7A	i2.8xlarge	32	244	7.625	6.82
ZJC9VZJF5NZNYSVK	d2.4xlarge	16	122	7.625	2.76
J4T9ZF4AJ2DXE7SA	m4.10xlarge	40	160	4	2.394
3RUU5T58T7XAFAAF	cr1.8xlarge	32	244	7.625	3.5
YGU2QZY8VPP94FSR	m3.large	2	7.5	3.75	0.133
4TCUDNKW7PMPSUT2	r3.8xlarge	32	244	7.625	2.66
MU4QGTJYWR6T73MZ	i2.2xlarge	8	61	7.625	1.705
HZC9FAP4F9Y8JW67	t2.micro	1	1	1	0.013
...					

Answer problem 1: instances with most RAM and most RAM per

Instances with the most RAM:

```
print("Instance with most RAM:")
mostRAM = max(row[3] for row in instances)
print(tabulate(filter(lambda row:row[3]==mostRAM,instances),
               headers,tablefmt="simple"))
```

```
row = (product,                    # row[0]
       patts['instanceType'],      # row[1]
       vcpu,                       # row[2]
       gb,                         # row[3]
       gb/vcpu,                    # row[4]
       float(dv['pricePerUnit']['USD'])) # row[5]
```

```
Instance with most RAM:
product           instanceType      vCPU    Memory    GiB/cpu    pricePerUnit
----------------  --------------  ------  --------  ---------  --------------
XP5P8NMSB2W7KP3U  d2.8xlarge         36       244    6.77778             5.52
3RUU5T58T7XAFAAF  cr1.8xlarge         32       244    7.625               3.5
NARXYND9H74FTC7A  i2.8xlarge         32       244    7.625               6.82
4TCUDNKW7PMPSUT2  r3.8xlarge         32       244    7.625               2.66
```

Instances with the most RAM per Core:

```
mostRAMPerCPU = max(row[4] for row in instances)
print(tabulate(filter(lambda row:row[4]==mostRAMPerCPU,instances),
               headers,tablefmt="simple"))
```

```
product           instanceType      vCPU    Memory    GiB/cpu    pricePerUnit
----------------  --------------  ------  --------  ---------  --------------
QCQ27AYFPSSTJG55  m2.2xlarge          4      34.2       8.55            0.49
J6U6GMEFVH686HBN  m2.xlarge           2      17.1       8.55            0.245
RKCQDTMY5DZS4JWT  m2.4xlarge          8      68.4       8.55            0.98
```

Debugging is an experimental science.

Simplify, Simplify, Simplify.

- If you have a three-step problem, debug step 1, then step 2, then step 3.
- Run "-r inline", then "-r local", then "-r hadoop"
  - *inline — runs the mapper and reducer in the same process.*
  - *local — runs the mapper and reducer in another process, communicates with a pipe. (like Hadoop Streaming)*
  - *hadoop — runs on multiple machines, multiple VMs, with Hadoop streaming.*

Always develop with a small data set.

- If it crashes with all of 2012:
  - *Try with one day — 2012-01-01*
  - *Try with another day — 2012-06-15*
  - *Try with a month*
  - *Try with one element of the maxmind data set.*

Take lots of notes. Write down what works and what doesn't work. Look at the output.

Create and test hypotheses. Make minor changes and see what happens.

Experiment on your code.

https://pixabay.com/en/learn-know-students-chalk-children-916677/

Student Presentations

Weiye Deng	Program	Google Cloud Environment
Zhengning Li	Paper	Spark SQL: Relational Data Processing in Spark
Tim Ahn	Program	Google BigQuery

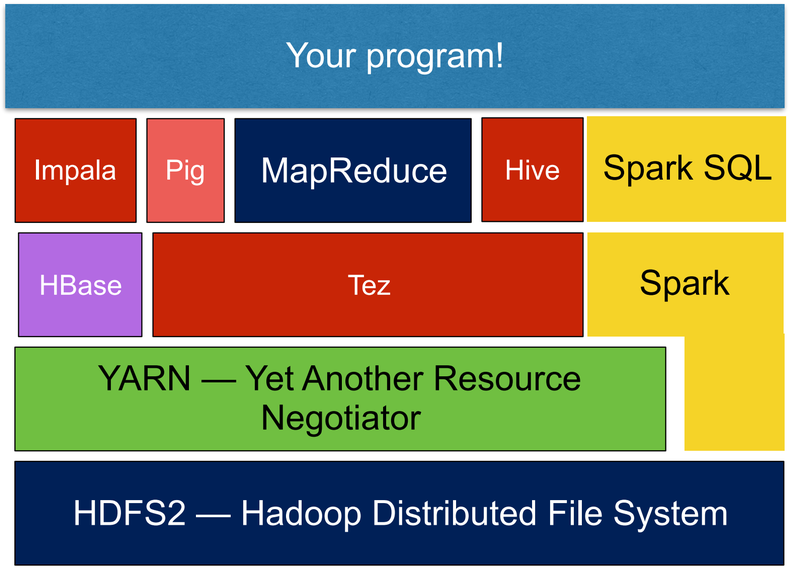
Introducing Spark

Spark — Not another layer on MapReduce

Fundamentally different computation model — Resilient Dynamic Datasets (RDDs)

Combines ideas from Pig and MapReduce, but in memory:

- Computes execution graph.
- Copies data from memory → memory (disk isn't used for intermediate results)
- Much faster than MapReduce — up to 40x faster



Compatible with Hadoop:

- Read/write any Hadoop data source — HDFS, HBase, SequenceFiles, etc.
- Runs with YARN — Share a cluster with MapReduce, Pig, etc.

Developed at UC Berkeley by the AMP Lab

- http://spark-project.org/

Spark — a new programming model for massive computing

MapReduce limitations:

- Designed for data flow, not for iterative calculations — e.g. Machine Learning
- Designed for batch processing — high overhead, slow.
- Java/JVM — other language require "streaming" interface

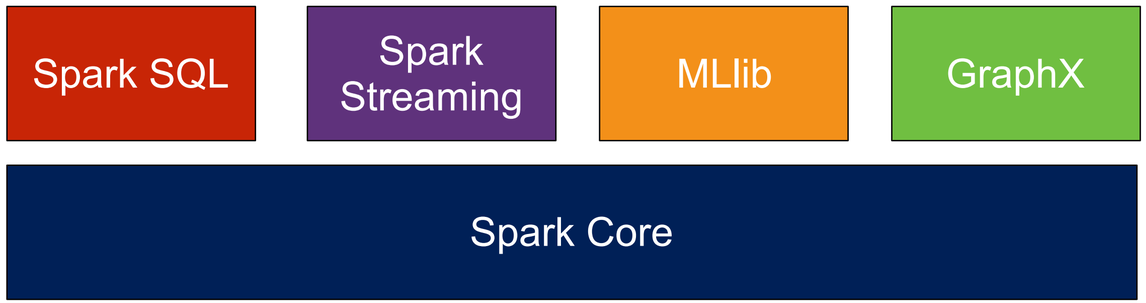
Spark:

- Designed for data sharing between steps — Iterative processing
- Designed to support interactive programming.
- Added support for streaming processing.
- Flexible, expressive programming model.

Four ways to run:

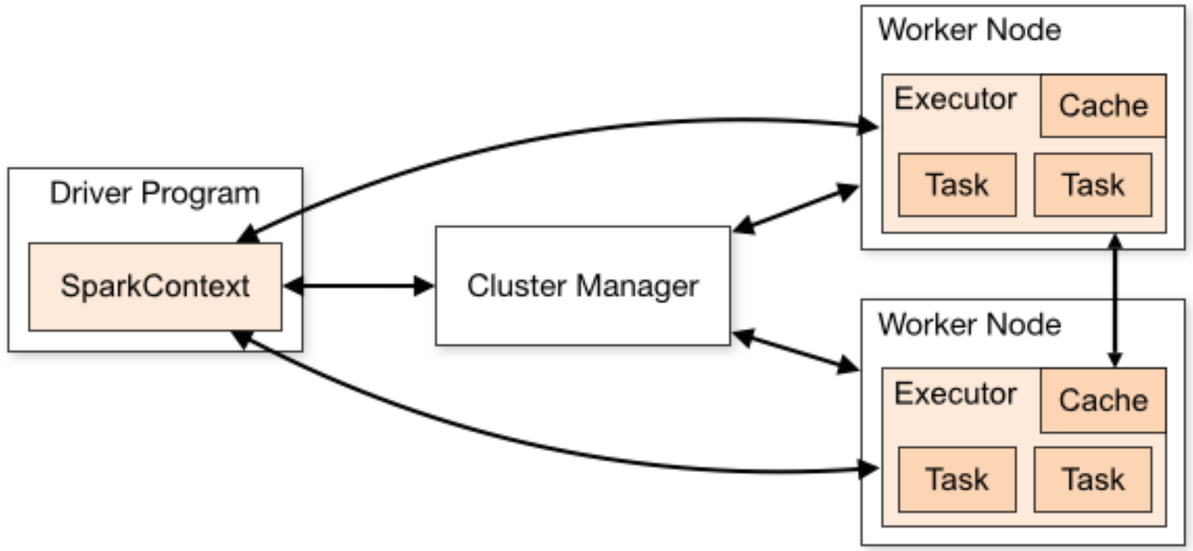
- Local Mode
- EC2 — multiple systems, native OS
- Apache Mesos (cluster management software)*
- Apache YARN

Key Spark Parts:



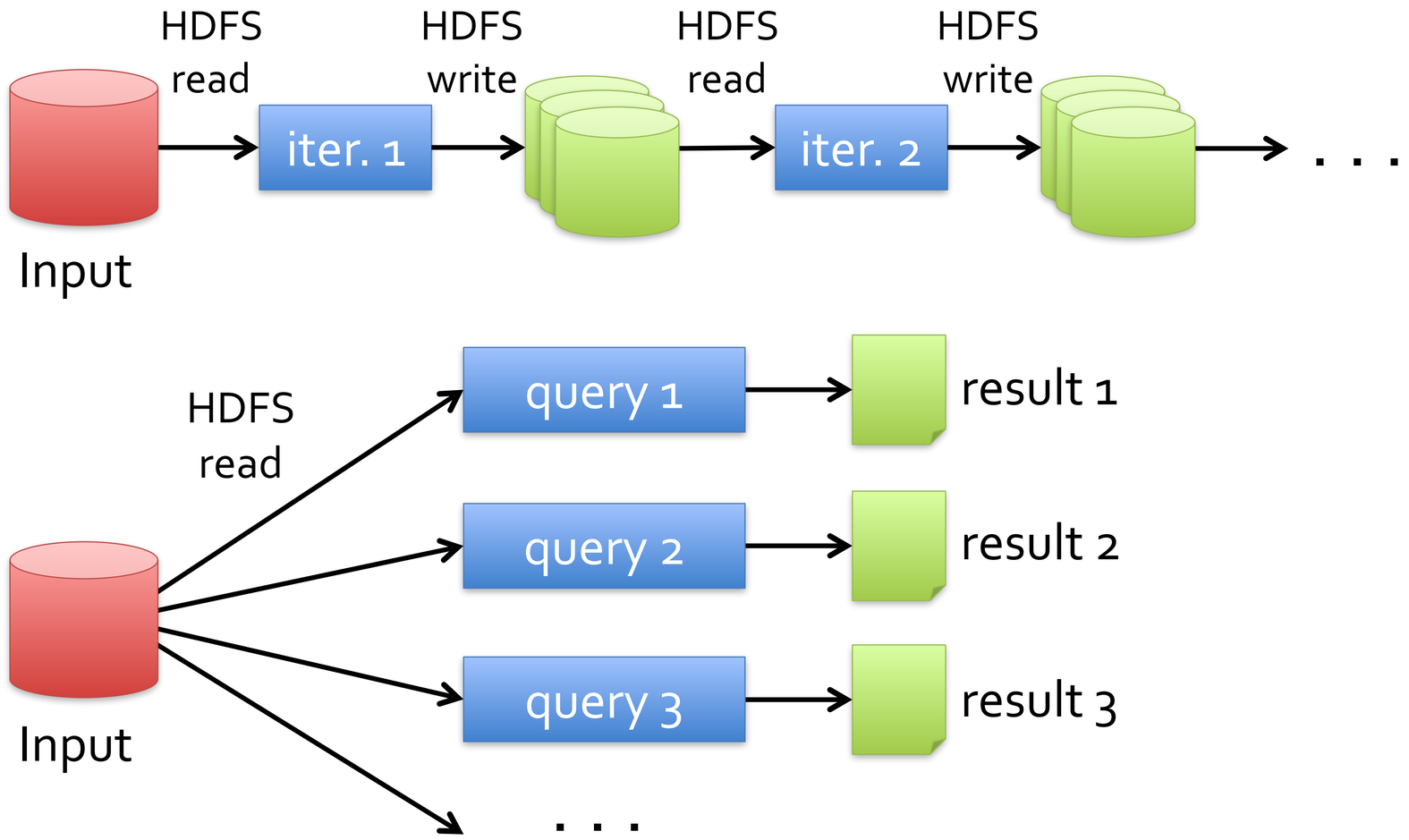
*https://aws.amazon.com/blogs/compute/cluster-management-with-amazon-ecs/

Spark Components

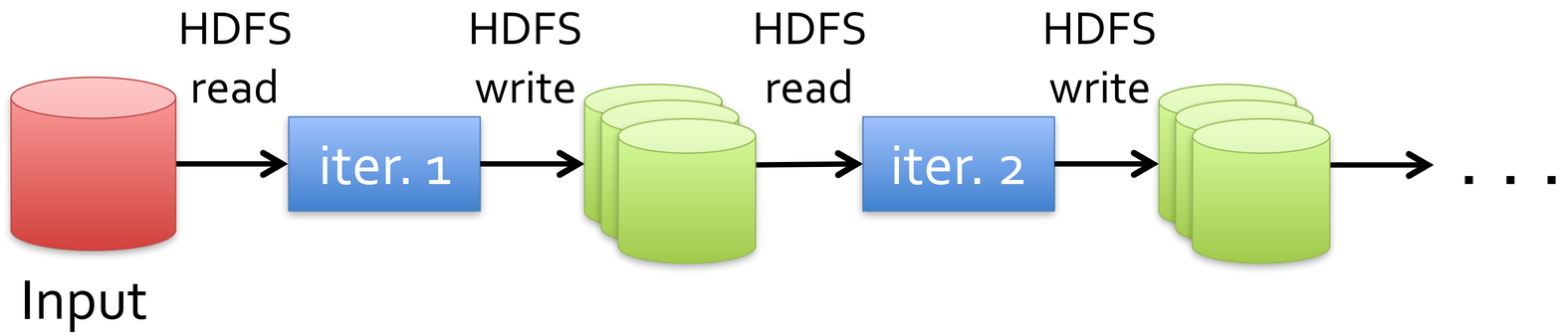


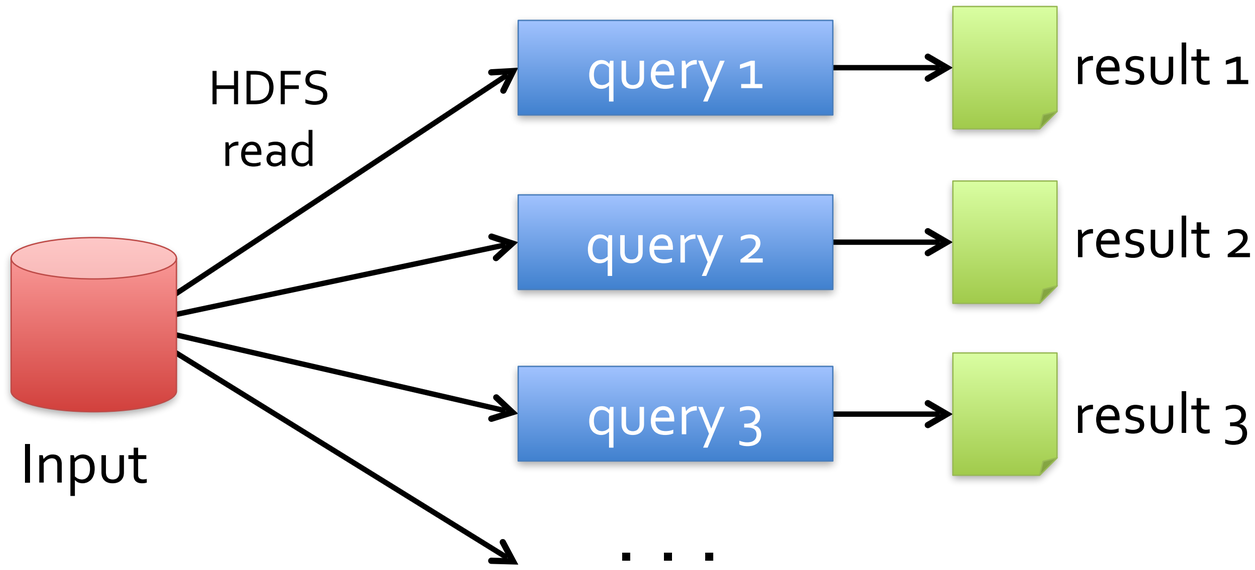
http://spark.apache.org/docs/latest/cluster-overview.html

Data Sharing in MapReduce



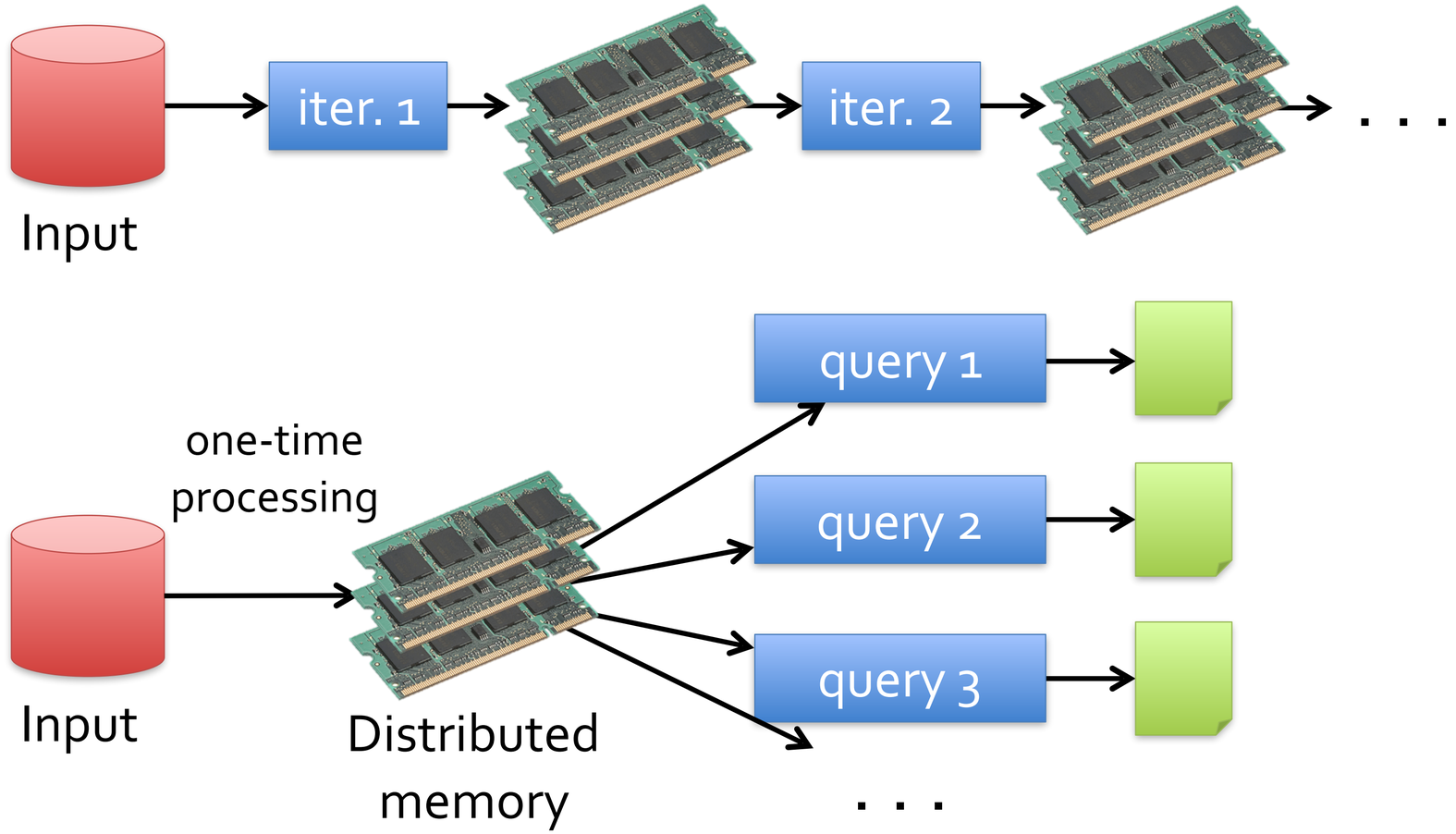
Data Sharing in MapReduce





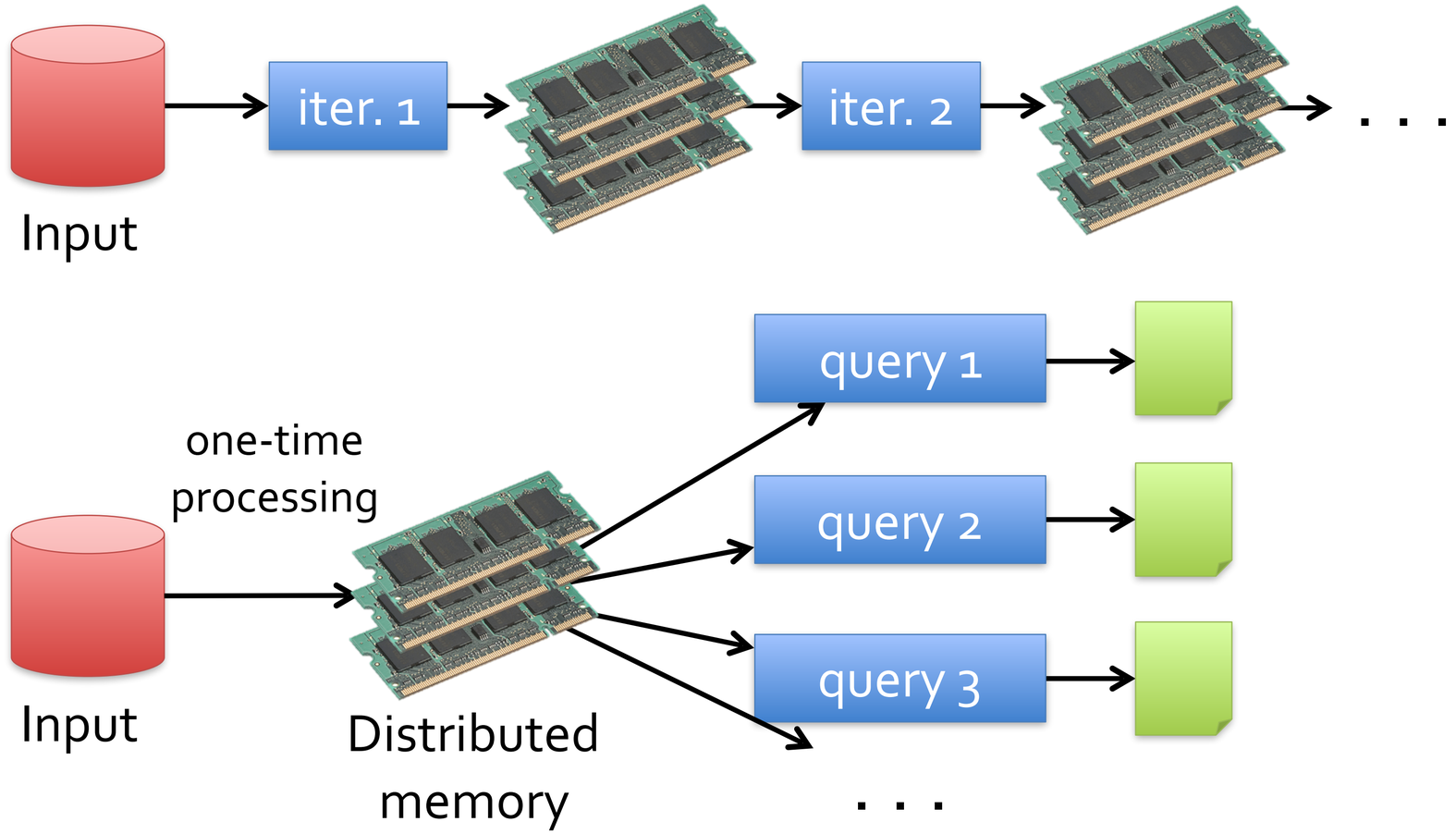
Slow due to replication, serialization, and disk IO

Data Sharing in Spark



Data Sharing in Spark

AMP Lab Slide



10-100× faster than network and disk

Spark programming model

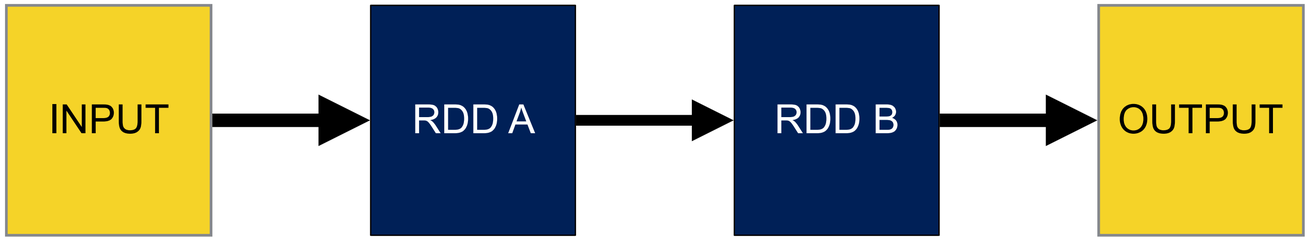
Key idea: resilient distributed datasets (RDDs)

- Holds structured data.
- Distributed across 1 or more nodes.
- Connected together in a directed acyclic graph
- Automatically rebuilt on failure

Working with RDDs:

- Manipulated with high-level interface.
- Read-Only (do not change after created)
- "Multiset" — distributed over multiple nodes.
- Interactive manipulation from:
  - *Scala*
  - *Python*
  - *R*

Data flows

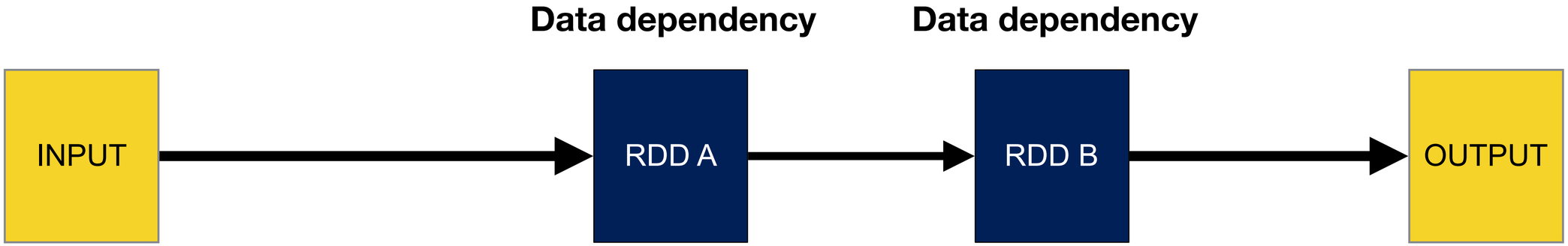


Spark requires:

- **Cluster manager**
- **Distributed storage system**

You write code in Python that performs a computation.

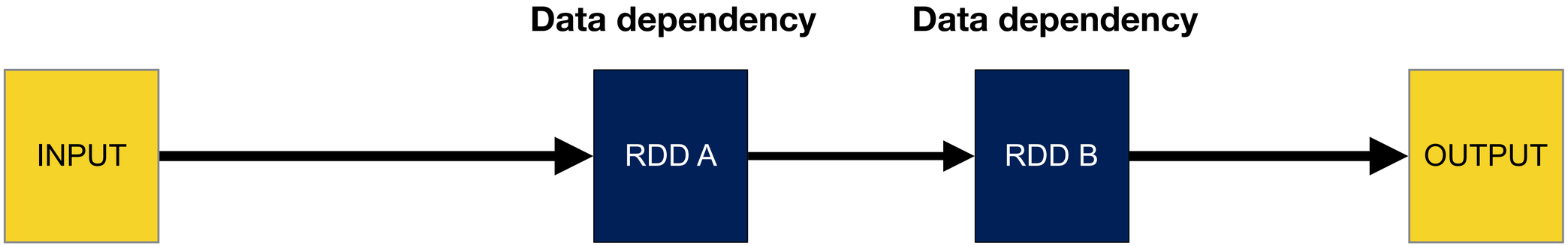
Sample Question: How many lines in Shakespeare contain the word "Hamlet" ?



s3://gu-anly502/ps04/Shakespeare.txt

You write code in Python that performs a computation.

Sample Question: How many lines in Shakespeare contain the word "Hamlet" ?



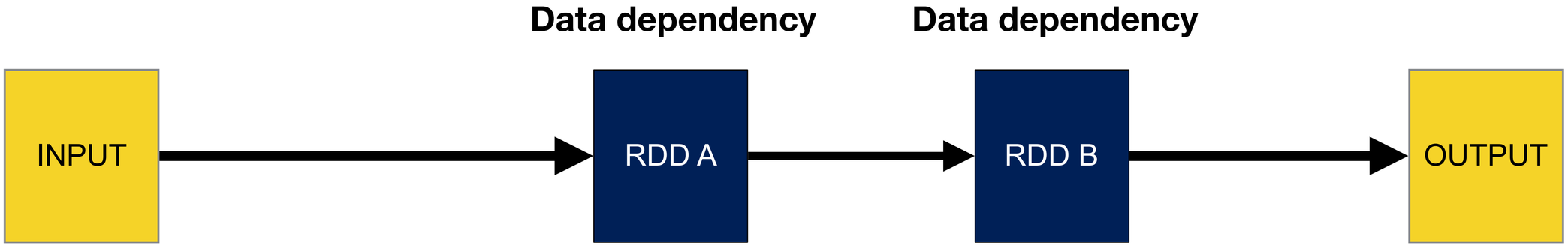
s3://gu-anly502/ps04/Shakespeare.txt

1. Read the file from S3 line-by-line

```
A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
```

You write code in Python that performs a computation.

Sample Question: How many lines in Shakespeare contain the word "Hamlet" ?



s3://gu-anly502/ps04/Shakespeare.txt

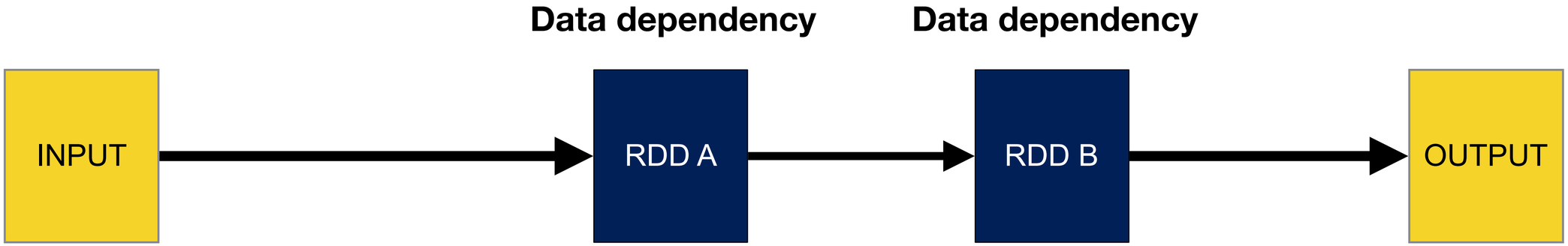
1. Read the file from S3 line-by-line

2. Filter the lines for "Hamlet"

```
A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")

def hasHamlet( s ):
    return "Hamlet" in s

B = A.filter( hasHamlet )
```

You write code in Python that performs a computation.

Sample Question: How many lines in Shakespeare contain the word "Hamlet" ?



s3://gu-anly502/ps04/Shakespeare.txt

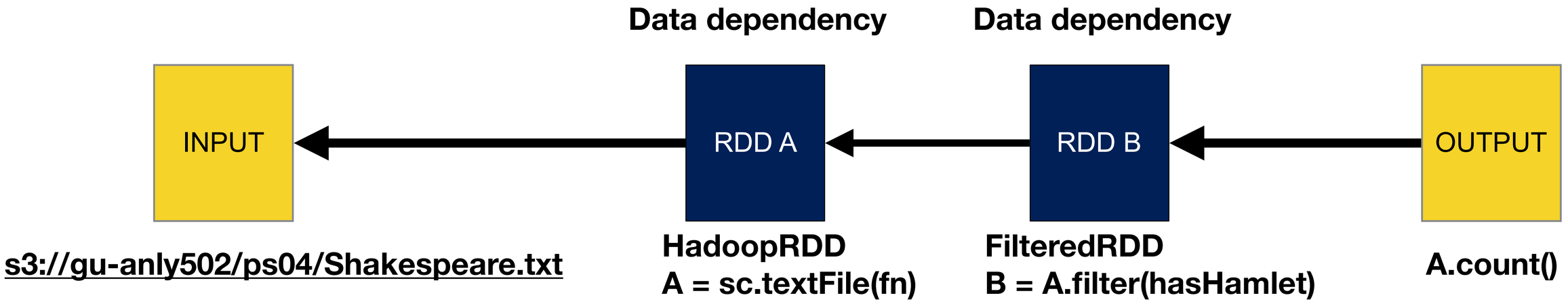
1. Read the file from S3 line-by-line

2. Filter the lines for "Hamlet"

3. Print Line Count

```
A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")

def hasHamlet( s ):
    return "Hamlet" in s

B = A.filter( hasHamlet )

print( B.count() )
```

Behind the scenes, Spark builds a dependency tree



When the output is requested, the RDDs are created as necessary.

DEMO
1. Start Spark (ipyspark on EMR using s3://gu-anly502/bootstrap.sh)

```
[15:57:16 last: 0s][~]
$ ipyspark
Python 2.7.10 (default, Dec  8 2015, 18:25:23)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
16/02/28 15:57:28 INFO SparkContext: Running Spark version 1.6.0
16/02/28 15:57:29 INFO SecurityManager: Changing view acls to: hadoop
16/02/28 15:57:29 INFO SecurityManager: Changing modify acls to: hadoop
16/02/28 15:57:29 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions:
Set(hadoop); users with modify permissions: Set(hadoop)
16/02/28 15:57:29 INFO Utils: Successfully started service 'sparkDriver' on port 43948.
16/02/28 15:57:30 INFO Slf4jLogger: Slf4jLogger started
16/02/28 15:57:30 INFO Remoting: Starting remoting
16/02/28 15:57:30 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@172.31.41.192:50951]
16/02/28 15:57:30 INFO Utils: Successfully started service 'sparkDriverActorSystem' on port 50951.
16/02/28 15:57:30 INFO SparkEnv: Registering MapOutputTracker
16/02/28 15:57:30 INFO SparkEnv: Registering BlockManagerMaster
16/02/28 15:57:30 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-e898815e-9f74-4563-8f20-d33412dcbc24
16/02/28 15:57:30 INFO MemoryStore: MemoryStore started with capacity 518.1 MB
16/02/28 15:57:31 INFO SparkEnv: Registering OutputCommitCoordinator
16/02/28 15:57:31 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
16/02/28 15:57:31 INFO Utils: Successfully started service 'SparkUI' on port 4041.
16/02/28 15:57:31 INFO SparkUI: Started SparkUI at http://172.31.41.192:4041
16/02/28 15:57:31 INFO RMProxy: Connecting to ResourceManager at ip-172-31-41-192.ec2.internal/172.31.41.192:8032
16/02/28 15:57:31 INFO Client: Requesting a new application from cluster with 1 NodeManagers
16/02/28 15:57:31 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster
(11520 MB per container)
...
```

DEMO
1. Start Spark (ipyspark on EMR using s3://gu-anly502/bootstrap.sh)

```
[15:57:16 last: 0s][~]
$ ipyspark
Python 2.7.10 (default, Dec  8 2015, 18:25:23)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
16/02/28 15:57:28 INFO SparkContext: Running Spark version 1.6.0
16/02/28 15:57:29 INFO SecurityManager: Changing view acls to: hadoop
16/02/28 15:57:29 INFO SecurityManager: Changing modify acls to: hadoop
16/02/28 15:57:29 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions:
Set(hadoop); users with modify permissions: Set(hadoop)
16/02/28 15:57:29 INFO Utils: Successfully started service 'sparkDriver' on port 43948.
16/02/28 15:57:30 INFO Slf4jLogger: Slf4jLogger started
16/02/28 15:57:30 INFO Remoting: Starting remoting
16/02/28 15:57:30 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@172.31.41.192:50951]
16/02/28 15:57:30 INFO Utils: Successfully started service 'sparkDriverActorSystem' on port 50951.
16/02/28 15:57:30 INFO SparkEnv: Registering MapOutputTracker
16/02/28 15:57:30 INFO SparkEnv: Registering BlockManagerMaster
16/02/28 15:57:30 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-e898815e-9f74-4563-8f20-d33412dcbc24
16/02/28 15:57:30 INFO MemoryStore: MemoryStore started with capacity 518.1 MB
16/02/28 15:57:31 INFO SparkEnv: Registering OutputCommitCoordinator
16/02/28 15:57:31 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
16/02/28 15:57:31 INFO Utils: Successfully started service 'SparkUI' on port 4041.
16/02/28 15:57:31 INFO SparkUI: Started SparkUI at http://172.31.41.192:4041
16/02/28 15:57:31 INFO RMProxy: Connecting to ResourceManager at ip-172-31-41-192.ec2.internal/172.31.41.192:8032
16/02/28 15:57:31 INFO Client: Requesting a new application from cluster with 1 NodeManagers
16/02/28 15:57:31 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster
(11520 MB per container)
...
```

New Prompt!

[Time of Day, Time of last command][Current Directory]

```
16/02/28 15:57:47 INFO BlockManagerMasterEndpoint: Registering block manager ip-172-31-41-192.ec2.internal:56815 with 518.1 MB RAM,
BlockManagerId(2, ip-172-31-41-192.ec2.internal, 56815)
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.7.10 (default, Dec  8 2015 18:25:23)
SparkContext available as sc, HiveContext available as sqlContext.

In [1]: A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
16/02/28 16:15:22 INFO MemoryStore: Block broadcast_0 stored as values in memory (estimated size 201.5 KB, free 201.5 KB)
16/02/28 16:15:23 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 24.0 KB, free 225.4 KB)
16/02/28 16:15:23 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on 172.31.41.192:34023 (size: 24.0 KB, free: 518.0 MB)
16/02/28 16:15:23 INFO SparkContext: Created broadcast 0 from textFile at NativeMethodAccessorImpl.java:-2

In [2]: def hasHamlet( s ):
   ...:     return "Hamlet" in s
   ...:

In [3]: B = A.filter( hasHamlet )

In [4]: print( B.count() )
```

```
In [4]: print( B.count() )
16/02/28 16:15:45 INFO GPLNativeCodeLoader: Loaded native gpl library
16/02/28 16:15:45 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev 02f444f0932ea7710dcc4bcdc1aa7ca55adf48c9]
16/02/28 16:15:45 INFO EmrFileSystem: Consistency disabled, using com.amazon.ws.emr.hadoop.fs.s3n.S3NativeFileSystem as filesystem implementation
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[null], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=0, ClientExecuteTime=[299.723], HttpRequestTime=[238.828],
HttpClientReceiveResponseTime=[17.905], RequestSigningTime=[19.15], CredentialsRequestTime=[0.021], ResponseProcessingTime=[0.822],
HttpClientSendRequestTime=[0.596],
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[null], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=1, ClientExecuteTime=[8.939], HttpRequestTime=[8.34],
HttpClientReceiveResponseTime=[5.69], RequestSigningTime=[0.281], CredentialsRequestTime=[0.004], ResponseProcessingTime=[0.006],
HttpClientSendRequestTime=[0.586],
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[F92236463B7E28A8], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=1, ClientExecuteTime=[28.624], HttpRequestTime=[19.346],
HttpClientReceiveResponseTime=[17.179], RequestSigningTime=[0.253], CredentialsRequestTime=[0.004], ResponseProcessingTime=[8.649],
HttpClientSendRequestTime=[0.553],
16/02/28 16:15:46 INFO FileInputFormat: Total input paths to process : 1
16/02/28 16:15:46 INFO SparkContext: Starting job: count at <ipython-input-4-3255572840a3>:1
16/02/28 16:15:46 INFO DAGScheduler: Got job 0 (count at <ipython-input-4-3255572840a3>:1) with 2 output partitions
16/02/28 16:15:46 INFO DAGScheduler: Final stage: ResultStage 0 (count at <ipython-input-4-3255572840a3>:1)
16/02/28 16:15:46 INFO DAGScheduler: Parents of final stage: List()
16/02/28 16:15:46 INFO DAGScheduler: Missing parents: List()
16/02/28 16:15:46 INFO DAGScheduler: Submitting ResultStage 0 (PythonRDD[2] at count at <ipython-input-4-3255572840a3>:1), which has no missing parents
16/02/28 16:15:46 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 5.9 KB, free 231.3 KB)
16/02/28 16:15:46 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 3.7 KB, free 235.0 KB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on 172.31.41.192:34023 (size: 3.7 KB, free: 518.0 MB)
16/02/28 16:15:46 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
16/02/28 16:15:46 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 0 (PythonRDD[2] at count at <ipython-input-4-3255572840a3>:1)
16/02/28 16:15:46 INFO YarnScheduler: Adding task set 0.0 with 2 tasks
16/02/28 16:15:46 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, ip-172-31-41-192.ec2.internal, partition 0,RACK_LOCAL, 2138 bytes)
16/02/28 16:15:46 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, ip-172-31-41-192.ec2.internal, partition 1,RACK_LOCAL, 2138 bytes)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on ip-172-31-41-192.ec2.internal:56815 (size: 3.7 KB, free: 518.1 MB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on ip-172-31-41-192.ec2.internal:58349 (size: 3.7 KB, free: 518.1 MB)
16/02/28 16:15:47 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on ip-172-31-41-192.ec2.internal:56815 (size: 24.0 KB, free: 518.0 MB)
16/02/28 16:15:47 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on ip-172-31-41-192.ec2.internal:58349 (size: 24.0 KB, free: 518.0 MB)
16/02/28 16:15:52 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 5780 ms on ip-172-31-41-192.ec2.internal (1/2)
16/02/28 16:15:52 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 5833 ms on ip-172-31-41-192.ec2.internal (2/2)
16/02/28 16:15:52 INFO DAGScheduler: ResultStage 0 (count at <ipython-input-4-3255572840a3>:1) finished in 5.862 s
16/02/28 16:15:52 INFO YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
16/02/28 16:15:52 INFO DAGScheduler: Job 0 finished: count at <ipython-input-4-3255572840a3>:1, took 5.949927 s
108
```

```
In [4]: print( B.count() )
16/02/28 16:15:45 INFO GPLNativeCodeLoader: Loaded native gpl library
16/02/28 16:15:45 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev 02f444f0932ea7710dcc4bcdc1aa7ca55adf48c9]
16/02/28 16:15:45 INFO EmrFileSystem: Consistency disabled, using com.amazon.ws.emr.hadoop.fs.s3n.S3NativeFileSystem as filesystem implementation
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[null], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=0, ClientExecuteTime=[299.723], HttpRequestTime=[238.828],
HttpClientReceiveResponseTime=[17.905], RequestSigningTime=[19.15], CredentialsRequestTime=[0.021], ResponseProcessingTime=[0.822],
HttpClientSendRequestTime=[0.596],
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[null], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=1, ClientExecuteTime=[8.939], HttpRequestTime=[8.34],
HttpClientReceiveResponseTime=[5.69], RequestSigningTime=[0.281], CredentialsRequestTime=[0.004], ResponseProcessingTime=[0.006],
HttpClientSendRequestTime=[0.586],
16/02/28 16:15:46 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[F92236463B7E28A8], ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com],
HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0, HttpClientPoolAvailableCount=1, ClientExecuteTime=[28.624], HttpRequestTime=[19.346],
HttpClientReceiveResponseTime=[17.179], RequestSigningTime=[0.253], CredentialsRequestTime=[0.004], ResponseProcessingTime=[8.649],
HttpClientSendRequestTime=[0.553],
16/02/28 16:15:46 INFO FileInputFormat: Total input paths to process : 1
16/02/28 16:15:46 INFO SparkContext: Starting job: count at <ipython-input-4-3255572840a3>:1
16/02/28 16:15:46 INFO DAGScheduler: Got job 0 (count at <ipython-input-4-3255572840a3>:1) with 2 output partitions
16/02/28 16:15:46 INFO DAGScheduler: Final stage: ResultStage 0 (count at <ipython-input-4-3255572840a3>:1)
16/02/28 16:15:46 INFO DAGScheduler: Parents of final stage: List()
16/02/28 16:15:46 INFO DAGScheduler: Missing parents: List()
16/02/28 16:15:46 INFO DAGScheduler: Submitting ResultStage 0 (PythonRDD[2] at count at <ipython-input-4-3255572840a3>:1), which has no missing parents
16/02/28 16:15:46 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 5.9 KB, free 231.3 KB)
16/02/28 16:15:46 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 3.7 KB, free 235.0 KB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on 172.31.41.192:34023 (size: 3.7 KB, free: 518.0 MB)
16/02/28 16:15:46 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
16/02/28 16:15:46 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 0 (PythonRDD[2] at count at <ipython-input-4-3255572840a3>:1)
16/02/28 16:15:46 INFO YarnScheduler: Adding task set 0.0 with 2 tasks
16/02/28 16:15:46 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, ip-172-31-41-192.ec2.internal, partition 0,RACK_LOCAL, 2138 bytes)
16/02/28 16:15:46 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, ip-172-31-41-192.ec2.internal, partition 1,RACK_LOCAL, 2138 bytes)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on ip-172-31-41-192.ec2.internal:56815 (size: 3.7 KB, free: 518.1 MB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on ip-172-31-41-192.ec2.internal:58349 (size: 3.7 KB, free: 518.1 MB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on ip-172-31-41-192.ec2.internal:56815 (size: 24.0 KB, free: 518.0 MB)
16/02/28 16:15:46 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on ip-172-31-41-192.ec2.internal:58349 (size: 24.0 KB, free: 518.0 MB)
16/02/28 16:15:52 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 5780 ms on ip-172-31-41-192.ec2.internal (1/2)
16/02/28 16:15:52 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 5833 ms on ip-172-31-41-192.ec2.internal (2/2)
16/02/28 16:15:52 INFO DAGScheduler: ResultStage 0 (count at <ipython-input-4-3255572840a3>:1) finished in 5.862 s
16/02/28 16:15:52 INFO YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
16/02/28 16:15:52 INFO DAGScheduler: Job 0 finished: count at <ipython-input-4-3255572840a3>:1, took 5.949927 s
108
```

With Spark, you can look at your data interactively!

```
In [1]: A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
In [2]: def hasHamlet( s ):
   ...:     return "Hamlet" in s
   ...:
In [3]: B = A.filter( hasHamlet )
```

```
In [5]: A.first()
Out[5]: u'<<THIS ELECTRONIC VERSION OF THE COMPLETE WORKS OF WILLIAM'


In [6]: B.first()
Out[6]: u'  Hamlet, son to the former, and nephew to the present king.'

In [7]: B.collect()
Out[7]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.',
 u"  Ghost of Hamlet's Father.",
 u"    Dar'd to the combat; in which our valiant Hamlet",
 u'    His fell to Hamlet. Now, sir, young Fortinbras,',
 u'    Unto young Hamlet; for, upon my life,',
...

In [8]: B.take(3)
Out[8]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.']

In [9]: B.takeSample(False,3)
Out[9]:
[u'    Hamlet comes back. What would you undertake',
 u'                                        Exeunt Ghost and Hamlet.',
 u'    last king Hamlet overcame Fortinbras.']
```

With Spark, you can look at your data interactively!

```
In [1]: A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
In [2]: def hasHamlet( s ):
   ...:     return "Hamlet" in s
   ...:
In [3]: B = A.filter( hasHamlet )
```

```
In [5]: A.first()
Out[5]: u'<<THIS ELECTRONIC VERSION OF THE COMPLETE WORKS OF WILLIAM'


In [6]: B.first()
Out[6]: u'  Hamlet, son to the former, and nephew to the present king.'

In [7]: B.collect()
Out[7]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.',
 u"  Ghost of Hamlet's Father.",
 u"    Dar'd to the combat; in which our valiant Hamlet",
 u'    His fell to Hamlet. Now, sir, young Fortinbras,',
 u'    Unto young Hamlet; for, upon my life,',
...

In [8]: B.take(3)
Out[8]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.']

In [9]: B.takeSample(False,3)
Out[9]:
[u'    Hamlet comes back. What would you undertake',
 u'                                        Exeunt Ghost and Hamlet.',
 u'    last king Hamlet overcame Fortinbras.']
```

With Spark, you can look at your data interactively!

```
In [1]: A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
In [2]: def hasHamlet( s ):
   ...:     return "Hamlet" in s
   ...:
In [3]: B = A.filter( hasHamlet )
```

```
In [5]: A.first()
Out[5]: u'<<THIS ELECTRONIC VERSION OF THE COMPLETE WORKS OF WILLIAM'


In [6]: B.first()
Out[6]: u'  Hamlet, son to the former, and nephew to the present king.'

In [7]: B.collect()
Out[7]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.',
 u"  Ghost of Hamlet's Father.",
 u"    Dar'd to the combat; in which our valiant Hamlet",
 u'    His fell to Hamlet. Now, sir, young Fortinbras,',
 u'    Unto young Hamlet; for, upon my life,',
...

In [8]: B.take(3)
Out[8]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.']

In [9]: B.takeSample(False,3)
Out[9]:
[u'    Hamlet comes back. What would you undertake',
 u'                                        Exeunt Ghost and Hamlet.',
 u'    last king Hamlet overcame Fortinbras.']
```

With Spark, you can look at your data interactively!

```
In [1]: A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
In [2]: def hasHamlet( s ):
   ...:     return "Hamlet" in s
   ...:
In [3]: B = A.filter( hasHamlet )
```

```
In [5]: A.first()
Out[5]: u'<<THIS ELECTRONIC VERSION OF THE COMPLETE WORKS OF WILLIAM'


In [6]: B.first()
Out[6]: u'  Hamlet, son to the former, and nephew to the present king.'

In [7]: B.collect()
Out[7]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.',
 u"  Ghost of Hamlet's Father.",
 u"    Dar'd to the combat; in which our valiant Hamlet",
 u'    His fell to Hamlet. Now, sir, young Fortinbras,',
 u'    Unto young Hamlet; for, upon my life,',
...

In [8]: B.take(3)
Out[8]:
[u'  Hamlet, son to the former, and nephew to the present king.',
 u'  Horatio, friend to Hamlet.',
 u'  Getrude, Queen of Denmark, mother to Hamlet.']

In [9]: B.takeSample(False,3)
Out[9]:
[u'    Hamlet comes back. What would you undertake',
 u'                                        Exeunt Ghost and Hamlet.',
 u'    last king Hamlet overcame Fortinbras.']
```

.first() takes first line

.take(n) takes first n

.takeSample(False,3) takes 3 random lines

Spark includes data sampling.
iPython provides interactive help.

Completion: Type "B.sam<*esc*><?>"

```
In [8]: B.take
B.take          B.takeOrdered   B.takeSample
```

Built-in help:

```
In [8]: help(B.takeSample)

   Help on method takeSample in module pyspark.rdd:

   takeSample(self, withReplacement, num, seed=None) method of pyspark.rdd.PipelinedRDD instance
       Return a fixed-size sampled subset of this RDD.

       >>> rdd = sc.parallelize(range(0, 10))
       >>> len(rdd.takeSample(True, 20, 1))
       20
       >>> len(rdd.takeSample(False, 5, 2))
       5
       >>> len(rdd.takeSample(False, 15, 3))
       10
   (END)
```

sample() — performs sampling into an RDD
takeSample() — Combines sampling and .collect()

```
sample(self, withReplacement, fraction, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a sampled subset of this RDD.

takeSample(self, withReplacement, num, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a fixed-size sampled subset of this RDD.
```

```
In [10]: B.sample(False,.10)
Out[10]: PythonRDD[7] at RDD at PythonRDD.scala:43

In [11]: B.sample(False,.10).collect()
...
Out[11]:
[u'    Unto young Hamlet; for, upon my life,',
 u'  Queen. Good Hamlet, cast thy nighted colour off,',
 u'  Oph. So please you, something touching the Lord Hamlet.  ',
 u'    Than a command to parley. For Lord Hamlet,',
 u"    Lord Hamlet, with his doublet all unbrac'd,",
 u"    Of Hamlet's transformation. So I call it,",
 u'                  [Exit the Queen. Then] Exit Hamlet, tugging in',
 u'        Enter Hamlet and Guildenstern [with Attendants].',
 u'  King. Hamlet, this deed, for thine especial safety,-',
 u'    The present death of Hamlet. Do it, England;  ',
 u'                                        [Exeunt all but Hamlet.]',
 u"    And that in Hamlet's hearing, for a quality",
 u'  King. Stay, give me drink. Hamlet, this pearl is thine;']

In [12]:
```

Execution time: 2 seconds

sample() — performs sampling into an RDD
takeSample() — Combines sampling and .collect()

```
sample(self, withReplacement, fraction, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a sampled subset of this RDD.

takeSample(self, withReplacement, num, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a fixed-size sampled subset of this RDD.
```

.sample() returns an RDD

```
In [10]: B.sample(False,.10)
Out[10]: PythonRDD[7] at RDD at PythonRDD.scala:43

In [11]: B.sample(False,.10).collect()
...
Out[11]:
[u'    Unto young Hamlet; for, upon my life,',
 u'  Queen. Good Hamlet, cast thy nighted colour off,',
 u'  Oph. So please you, something touching the Lord Hamlet.  ',
 u'    Than a command to parley. For Lord Hamlet,',
 u"    Lord Hamlet, with his doublet all unbrac'd,",
 u"    Of Hamlet's transformation. So I call it,",
 u'                  [Exit the Queen. Then] Exit Hamlet, tugging in',
 u'        Enter Hamlet and Guildenstern [with Attendants].',
 u'  King. Hamlet, this deed, for thine especial safety,-',
 u'    The present death of Hamlet. Do it, England;  ',
 u'                                        [Exeunt all but Hamlet.]',
 u"    And that in Hamlet's hearing, for a quality",
 u'  King. Stay, give me drink. Hamlet, this pearl is thine;']

In [12]:
```

Execution time: 2 seconds

sample() — performs sampling into an RDD
takeSample() — Combines sampling and .collect()

```
sample(self, withReplacement, fraction, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a sampled subset of this RDD.

takeSample(self, withReplacement, num, seed=None) method of pyspark.rdd.PipelinedRDD instance
    Return a fixed-size sampled subset of this RDD
```

.sample() returns an RDD

```
In [10]: B.sample(False,.10)
Out[10]: PythonRDD[7] at RDD at PythonRDD.scala:43

In [11]: B.sample(False,.10).collect()
...
Out[11]:
[u'    Unto young Hamlet; for, upon my life,',
 u'  Queen. Good Hamlet, cast thy nighted colour off,',
 u'  Oph. So please you, something touching the Lord Hamlet.  ',
 u'    Than a command to parley. For Lord Hamlet,',
 u"    Lord Hamlet, with his doublet all unbrac'd,",
 u"    Of Hamlet's transformation. So I call it,",
 u'                  [Exit the Queen. Then] Exit Hamlet, tugging in',
 u'        Enter Hamlet and Guildenstern [with Attendants].',
 u'  King. Hamlet, this deed, for thine especial safety,-',
 u'    The present death of Hamlet. Do it, England;  ',
 u'                                        [Exeunt all but Hamlet.]',
 u"    And that in Hamlet's hearing, for a quality",
 u'  King. Stay, give me drink. Hamlet, this pearl is thine;']

In [12]:
```

.collect() transfers the data to the driver

Execution time: 2 seconds

Same example, bigger dataset:

Question: How many times does Main_Page appear in the forensicswiki logs?

```
In [14]: s = sc.textFile("s3://gu-anly502/ps03/forensicswiki.2012.txt")
In [15]: smain = s.filter( lambda line:"Main_Page" in line )
In [16]: smain.count()
16/02/28 16:43:49 INFO latency: StatusCode=[200], ServiceName=[Amazon S3], AWSRequestID=[1CCBC4BAA1AC5655],
ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com], HttpClientPoolLeasedCount=0, RequestCount=1, HttpClientPoolPendingCount=0,
HttpClientPoolAvailableCount=0, ClientExecuteTime=[142.182], HttpRequestTime=[141.012], HttpClientReceiveResponseTime=[93.347],
RequestSigningTime=[0.331], CredentialsRequestTime=[0.005], ResponseProcessingTime=[0.172], HttpClientSendRequestTime=[0.541],
16/02/28 16:43:49 INFO FileInputFormat: Total input paths to process : 1
16/02/28 16:43:49 INFO SparkContext: Starting job: count at <ipython-input-16-b203ddda2853>:1
16/02/28 16:43:49 INFO DAGScheduler: Got job 7 (count at <ipython-input-16-b203ddda2853>:1) with 64 output partitions
16/02/28 16:43:49 INFO DAGScheduler: Final stage: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1)
16/02/28 16:43:49 INFO DAGScheduler: Parents of final stage: List()
16/02/28 16:43:49 INFO DAGScheduler: Missing parents: List()
16/02/28 16:43:49 INFO DAGScheduler: Submitting ResultStage 7 (PythonRDD[12] at count at <ipython-input-16-b203ddda2853>:1), which
has no missing parents
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9 stored as values in memory (estimated size 5.9 KB, free 441.0 KB)
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9_piece0 stored as bytes in memory (estimated size 3.7 KB, free 444.7 KB)
     ...
16/02/28 16:45:17 INFO TaskSetManager: Finished task 62.0 in stage 7.0 (TID 73) in 2699 ms on ip-172-31-41-192.ec2.internal (62/64)
16/02/28 16:45:17 INFO TaskSetManager: Finished task 61.0 in stage 7.0 (TID 72) in 2970 ms on ip-172-31-41-192.ec2.internal (63/64)
16/02/28 16:45:18 INFO TaskSetManager: Finished task 63.0 in stage 7.0 (TID 74) in 1289 ms on ip-172-31-41-192.ec2.internal (64/64)
16/02/28 16:45:18 INFO YarnScheduler: Removed TaskSet 7.0, whose tasks have all completed, from pool
16/02/28 16:45:18 INFO DAGScheduler: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1) finished in 89.607 s
16/02/28 16:45:18 INFO DAGScheduler: Job 7 finished: count at <ipython-input-16-b203ddda2853>:1, took 89.635715 s
Out[16]: 348820

In [17]:
```

Same example, bigger dataset:

Question: How many times does Main_Page appea[illegible]n the forensicswiki logs?

lambda notation!

```
In [14]: s = sc.textFile("s3://gu-anly502/ps03/forensicswiki.201
In [15]: smain = s.filter( lambda line:"Main_Page" in line )
In [16]: smain.count()
16/02/28 16:43:49 INFO latency: StatusCode=[200], ServiceName=[Ama[illegible]S3], AWSRequestID=[1CCBC4BAA1AC5655],
ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com], HttpClientPo[illegible]easedCount=0, RequestCount=1, HttpClientPoolPendingCount=0,
HttpClientPoolAvailableCount=0, ClientExecuteTime=[142.182], HttpRequestTime=[141.012], HttpClientReceiveResponseTime=[93.347],
RequestSigningTime=[0.331], CredentialsRequestTime=[0.005], ResponseProcessingTime=[0.172], HttpClientSendRequestTime=[0.541],
16/02/28 16:43:49 INFO FileInputFormat: Total input paths to process : 1
16/02/28 16:43:49 INFO SparkContext: Starting job: count at <ipython-input-16-b203ddda2853>:1
16/02/28 16:43:49 INFO DAGScheduler: Got job 7 (count at <ipython-input-16-b203ddda2853>:1) with 64 output partitions
16/02/28 16:43:49 INFO DAGScheduler: Final stage: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1)
16/02/28 16:43:49 INFO DAGScheduler: Parents of final stage: List()
16/02/28 16:43:49 INFO DAGScheduler: Missing parents: List()
16/02/28 16:43:49 INFO DAGScheduler: Submitting ResultStage 7 (PythonRDD[12] at count at <ipython-input-16-b203ddda2853>:1), which
has no missing parents
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9 stored as values in memory (estimated size 5.9 KB, free 441.0 KB)
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9_piece0 stored as bytes in memory (estimated size 3.7 KB, free 444.7 KB)
     ...
16/02/28 16:45:17 INFO TaskSetManager: Finished task 62.0 in stage 7.0 (TID 73) in 2699 ms on ip-172-31-41-192.ec2.internal (62/64)
16/02/28 16:45:17 INFO TaskSetManager: Finished task 61.0 in stage 7.0 (TID 72) in 2970 ms on ip-172-31-41-192.ec2.internal (63/64)
16/02/28 16:45:18 INFO TaskSetManager: Finished task 63.0 in stage 7.0 (TID 74) in 1289 ms on ip-172-31-41-192.ec2.internal (64/64)
16/02/28 16:45:18 INFO YarnScheduler: Removed TaskSet 7.0, whose tasks have all completed, from pool
16/02/28 16:45:18 INFO DAGScheduler: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1) finished in 89.607 s
16/02/28 16:45:18 INFO DAGScheduler: Job 7 finished: count at <ipython-input-16-b203ddda2853>:1, took 89.635715 s
Out[16]: 348820

In [17]:
```

Same example, bigger dataset:

Question: How many times does Main_Page appea[illegible]n the forensicswiki logs?

```
In [14]: s = sc.textFile("s3://gu-anly502/ps03/forensicswiki.201
In [15]: smain = s.filter( lambda line:"Main_Page" in line )
In [16]: smain.count()
16/02/28 16:43:49 INFO latency: StatusCode=[200], ServiceName=[Ama[illegible]S3], AWSRequestID=[1CCBC4BAA1AC5655],
ServiceEndpoint=[https://gu-anly502.s3.amazonaws.com], HttpClientPo[illegible]easedCount=0, RequestCount=1, HttpClientPoolPendingCount=0,
HttpClientPoolAvailableCount=0, ClientExecuteTime=[142.182], HttpRequestTime=[141.012], HttpClientReceiveResponseTime=[93.347],
RequestSigningTime=[0.331], CredentialsRequestTime=[0.005], ResponseProcessingTime=[0.172], HttpClientSendRequestTime=[0.541],
16/02/28 16:43:49 INFO FileInputFormat: Total input paths to process : 1
16/02/28 16:43:49 INFO SparkContext: Starting job: count at <ipython-input-16-b203ddda2853>:1
16/02/28 16:43:49 INFO DAGScheduler: Got job 7 (count at <ipython-input-16-b203ddda2853>:1) with 64 output partitions
16/02/28 16:43:49 INFO DAGScheduler: Final stage: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1)
16/02/28 16:43:49 INFO DAGScheduler: Parents of final stage: List()
16/02/28 16:43:49 INFO DAGScheduler: Missing parents: List()
16/02/28 16:43:49 INFO DAGScheduler: Submitting ResultStage 7 (PythonRDD[12] at count at <ipython-input-16-b203ddda2853>:1), which
has no missing parents
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9 stored as values in memory (estimated size 5.9 KB, free 441.0 KB)
16/02/28 16:43:49 INFO MemoryStore: Block broadcast_9_piece0 stored as bytes in memory (estimated size 3.7 KB, free 444.7 KB)
     ...
16/02/28 16:45:17 INFO TaskSetManager: Finished task 62.0 in stage 7.0 (TID 73) in 2699 ms on ip-172-31-41-192.ec2.internal (62/64)
16/02/28 16:45:17 INFO TaskSetManager: Finished task 61.0 in stage 7.0 (TID 72) in 2970 ms on ip-172-31-41-192.ec2.internal (63/64)
16/02/28 16:45:18 INFO TaskSetManager: Finished task 63.0 in stage 7.0 (TID 74) in 1289 ms on ip-172-31-41-192.ec2.intern[illegible] (64/64)
16/02/28 16:45:18 INFO YarnScheduler: Removed TaskSet 7.0, whose tasks have all completed, from pool
16/02/28 16:45:18 INFO DAGScheduler: ResultStage 7 (count at <ipython-input-16-b203ddda2853>:1) finished in 89.607 s
16/02/28 16:45:18 INFO DAGScheduler: Job 7 finished: count at <ipython-input-16-b203ddda2853>:1, took 89.635715 s
Out[16]: 348820

In [17]:
```

lambda notation!

89.6 sec!

With Spark, it's easy to check your work.

How do we trust this number:

```
In [16]: smain.count()
Out[16]: 348820

In [17]:
```

Try looking at some of the output:

```
In [17]: smain.sample(False,.001).collect()
Out[17]:
[u'193.105.210.94 - - [01/Jan/2012:08:35:04 -0800] "GET //Talk:Main_Page HTTP/1.0" 404 518 "http://www.forensicswiki.org//
Talk:Main_Page" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; APC; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR
2.0.50215; InfoPath.1)" ',
 u'32.178.74.29 - - [02/Jan/2012:12:18:16 -0800] "HEAD /index.php?title=Main_Page HTTP/1.1" 200 321 "-" "Mozilla/4.0
(compatible; Powermarks/3.5; Windows 95/98/2000/NT)" ',
 u'91.117.143.86 - - [02/Jan/2012:15:57:09 -0800] "GET /w/extensions/BibTex/bibtex.js HTTP/1.1" 200 1384 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (X11; U; i686 Linux; es, gl, en_GB, en_US) AppleWebKit/533.3 (KHTML, like
Gecko) Chrome/5.0.358.0 Safari/533.3" ',
 u'61.68.18.158 - - [03/Jan/2012:14:44:07 -0800] "GET /w/skins/monobook/main.css?270 HTTP/1.1" 304 174 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0.1) Gecko/20100101 Firefox/8.0.1" ',
 u'198.234.82.254 - - [04/Jan/2012:06:48:42 -0800] "GET /wiki/Tools HTTP/1.1" 200 13302 "http://www.forensicswiki.org/wiki/
Main_Page" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MDDR; InfoPath.2)" ',
...
```

With Spark, it's easy to check your work.

How do we trust this number:

```
In [16]: smain.count()
Out[16]: 348820

In [17]:
```

Try looking at some of the output:

```
In [17]: smain.sample(False,.001).collect()
Out[17]:
[u'193.105.210.94 - - [01/Jan/2012:08:35:04 -0800] "GET //Talk:Main_Page HTTP/1.0" 404 518 "http://www.forensicswiki.org//
Talk:Main_Page" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; APC; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR
2.0.50215; InfoPath.1)" ',
 u'32.178.74.29 - - [02/Jan/2012:12:18:16 -0800] "HEAD /index.php?title=Main_Page HTTP/1.1" 200 321 "-" "Mozilla/4.0
(compatible; Powermarks/3.5; Windows 95/98/2000/NT)" ',
 u'91.117.143.86 - - [02/Jan/2012:15:57:09 -0800] "GET /w/extensions/BibTex/bibtex.js HTTP/1.1" 200 1384 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (X11; U; i686 Linux; es, gl, en_GB, en_US) AppleWebKit/533.3 (KHTML, like
Gecko) Chrome/5.0.358.0 Safari/533.3" ',
 u'61.68.18.158 - - [03/Jan/2012:14:44:07 -0800] "GET /w/skins/monobook/main.css?270 HTTP/1.1" 304 174 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0.1) Gecko/20100101 Firefox/8.0.1" ',
 u'198.234.82.254 - - [04/Jan/2012:06:48:42 -0800] "GET /wiki/Tools HTTP/1.1" 200 13302 "http://www.forensicswiki.org/wiki/
Main_Page" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MDDR; InfoPath.2)" ',
...
```

Whoops! — We got the answer to our question, but it was the wrong question...

With Spark, it's easy to check your work.

How do we trust this number:

```
In [16]: smain.count()
Out[16]: 348820

In [17]:
```

Try looking at some of the output:

```
In [17]: smain.sample(False,.001).collect()
Out[17]:
[u'193.105.210.94 - - [01/Jan/2012:08:35:04 -0800] "GET //Talk:Main_Page HTTP/1.0" 404 518 "http://www.forensicswiki.org//
Talk:Main_Page" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; APC; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR
2.0.50215; InfoPath.1)" ',
 u'32.178.74.29 - - [02/Jan/2012:12:18:16 -0800] "HEAD /index.php?title=Main_Page HTTP/1.1" 200 321 "-" "Mozilla/4.0
(compatible; Powermarks/3.5; Windows 95/98/2000/NT)" ',
 u'91.117.143.86 - - [02/Jan/2012:15:57:09 -0800] "GET /w/extensions/BibTex/bibtex.js HTTP/1.1" 200 1384 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (X11; U; i686 Linux; es, gl, en_GB, en_US) AppleWebKit/533.3 (KHTML, like
Gecko) Chrome/5.0.358.0 Safari/533.3" ',
 u'61.68.18.158 - - [03/Jan/2012:14:44:07 -0800] "GET /w/skins/monobook/main.css?270 HTTP/1.1" 304 174 "http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0.1) Gecko/20100101 Firefox/8.0.1" ',
 u'198.234.82.254 - - [04/Jan/2012:06:48:42 -0800] "GET /wiki/Tools HTTP/1.1" 200 13302 "http://www.forensicswiki.org/wiki/
Main_Page" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MDDR; InfoPath.2)" ',
...
```

Whoops! — We got the answer to our question, but it was the wrong question...

With Spark, it's easy to check your work.

How do we trust this number:

```
In [16]: smain.count()
Out[16]: 348820

In [17]:
```

Try looking at some of the output:

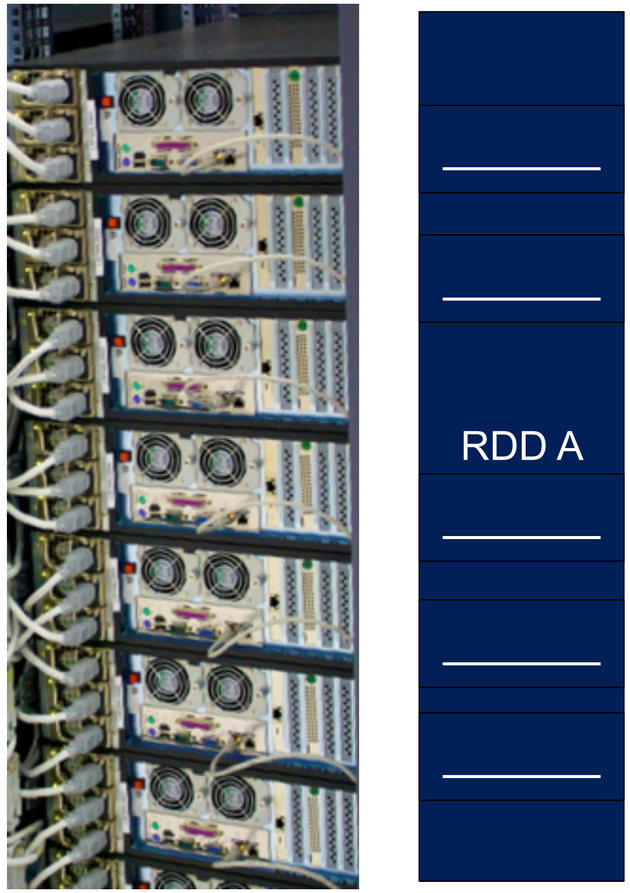
```
In [17]: smain.sample(False,.001).collect()
Out[17]:
[u'193.105.210.94 - - [01/Jan/2012:08:35:04 -0800] "GET //Talk:Main_Page HTTP/1.0" 404 518 "http://www.forensicswiki.org//
Talk:Main_Page" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; APC; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR
2.0.50215; InfoPath.1)" ',
 u'32.178.74.29 - - [02/Jan/2012:12:18:16 -0800] "HEAD /index.php?title=Main_Page HTTP/1.1" 200 321 "-" "Mozilla/4.0
(compatible; Powermarks/3.5; Windows 95/98/2000/NT)" ',
 u'91.117.143.86 - - [02/Jan/2012:15:57:09 -0800] "GET /w/extensions/BibTex/bi                                    http://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (X11; U; i686 Linux; es, gl                                    533.3 (KHTML, like
Gecko) Chrome/5.0.358.0 Safari/533.3" ',
 u'61.68.18.158 - - [03/Jan/2012:14:44:07 -0800] "GET /w/skins/monobook/main.c                                    tp://
www.forensicswiki.org/wiki/Main_Page" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0.1) Gecko/20100101 Firefox/8.0.1" ',
 u'198.234.82.254 - - [04/Jan/2012:06:48:42 -0800] "GET /wiki/Tools HTTP/1.1" 200 13302 "http://www.forensicswiki.org/wiki/
Main_Page" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MDDR; InfoPath.2)" ',
...
```

We'll solve this later!

Whoops! — We got the answer to our question, but it was the wrong question...

More about RDDs

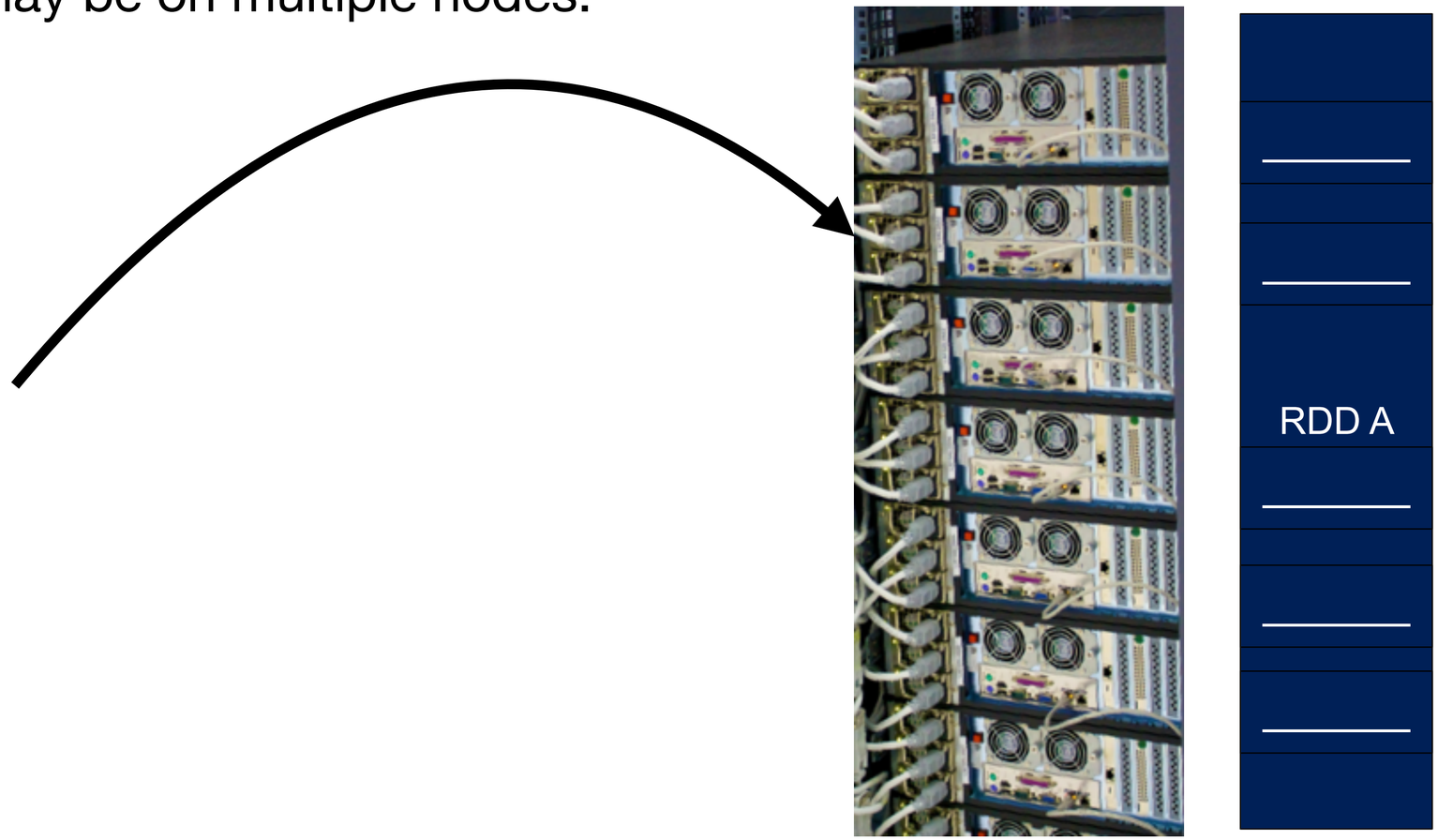
A RDD may be on multiple nodes.



HadoopRDD
A = sc.textFile(fn)

More about RDDs

A RDD may be on multiple nodes.

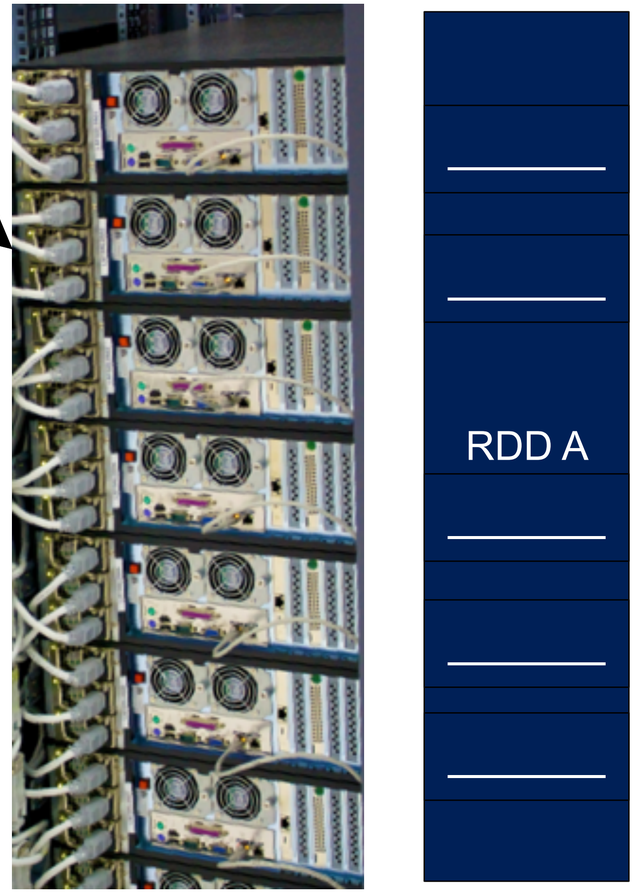


HadoopRDD
A = sc.textFile(fn)

More about RDDs

A RDD may be on multiple nodes.

The control program is called the "Driver."

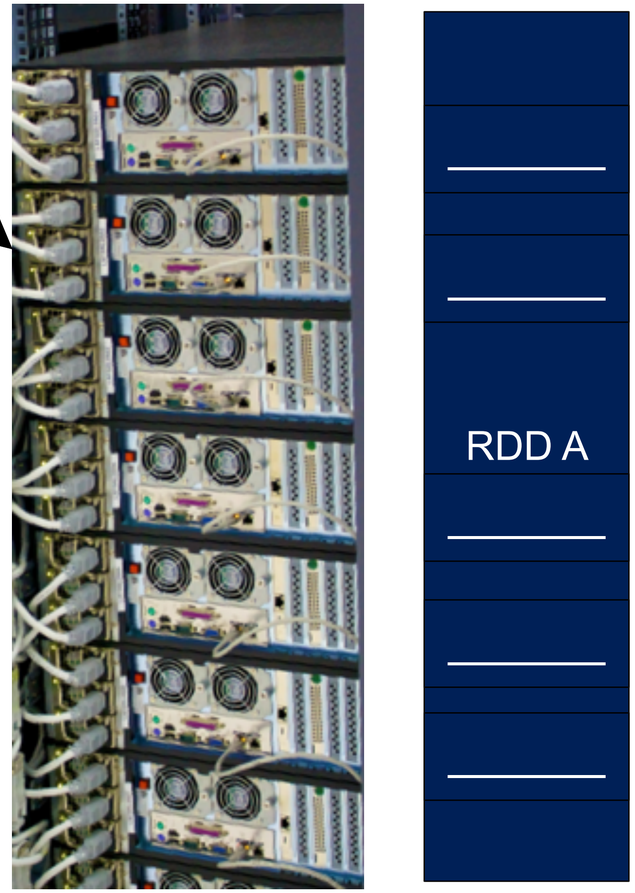


HadoopRDD
A = sc.textFile(fn)

More about RDDs

A RDD may be on multiple nodes.

The control program is called the "Driver."



HadoopRDD
A = sc.textFile(fn)

More about RDDs

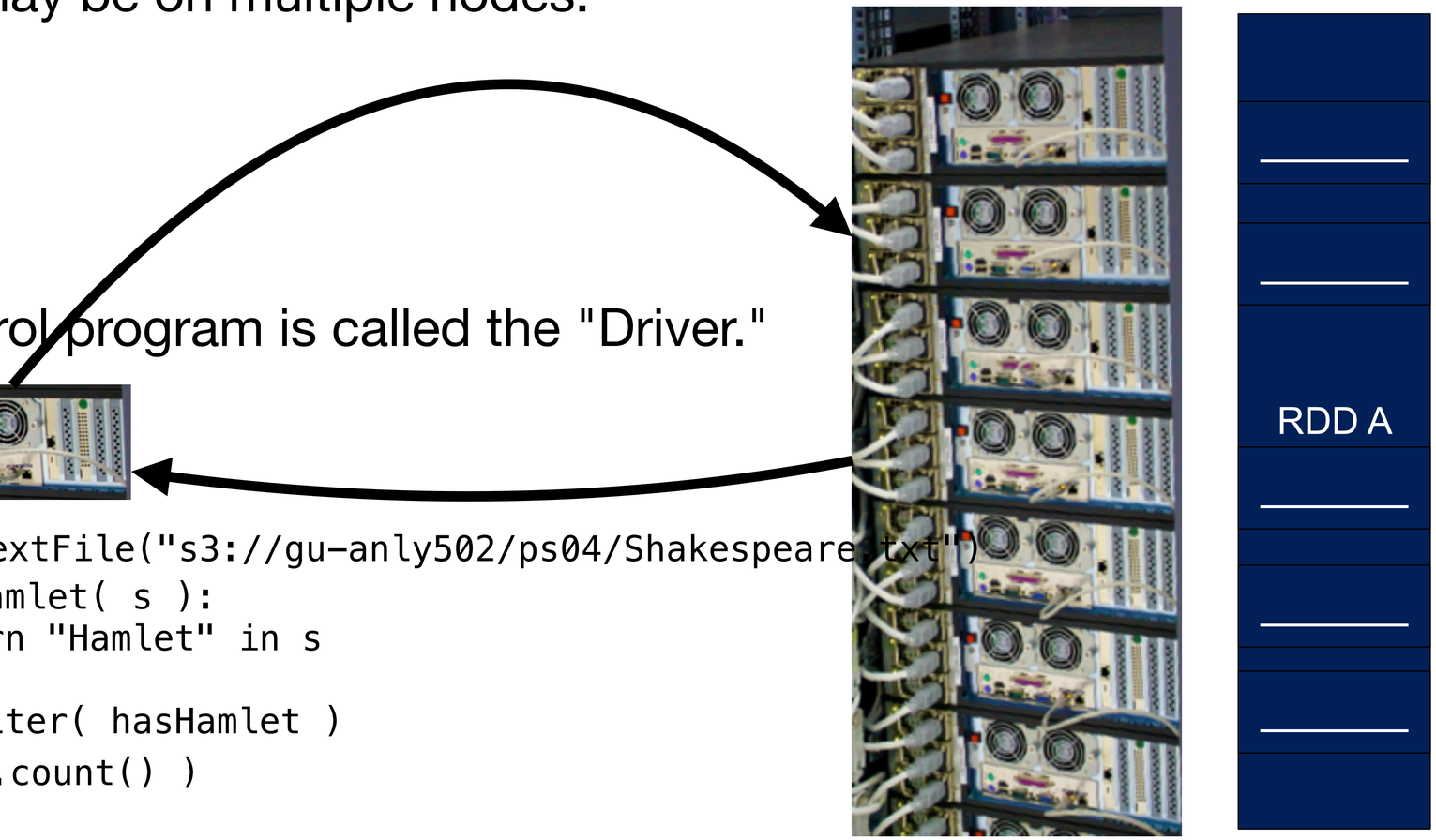
A RDD may be on multiple nodes.

The control program is called the "Driver."

HadoopRDD
A = sc.textFile(fn)

More about RDDs

A RDD may be on multiple nodes.

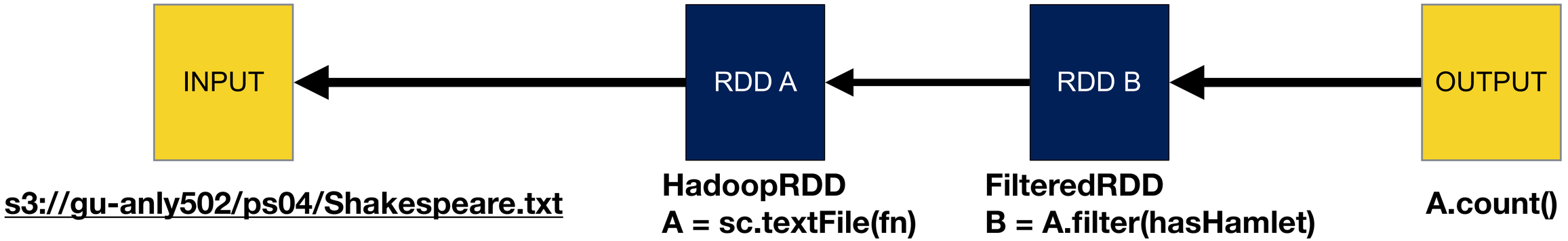


The control program is called the "Driver."

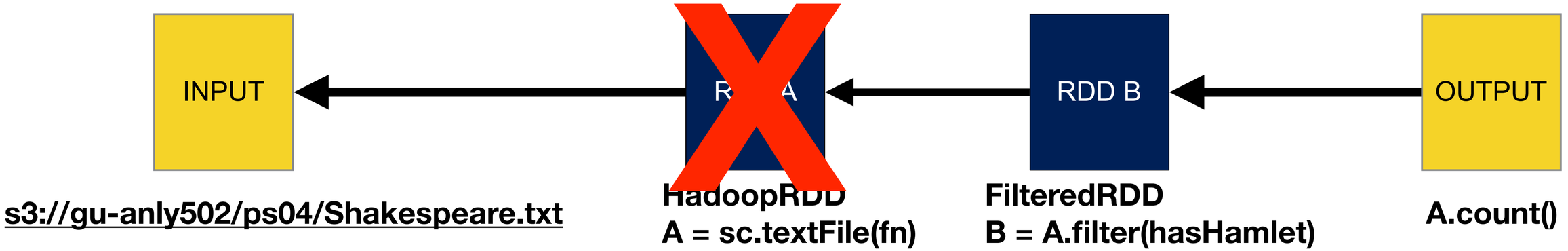
```
A = sc.textFile("s3://gu-anly502/ps04/Shakespeare.txt")
def hasHamlet( s ):
    return "Hamlet" in s

B = A.filter( hasHamlet )
print( B.count() )
```

HadoopRDD
A = sc.textFile(fn)

RDDs are deterministic.
If an RDD fails, it is rebuilt automatically

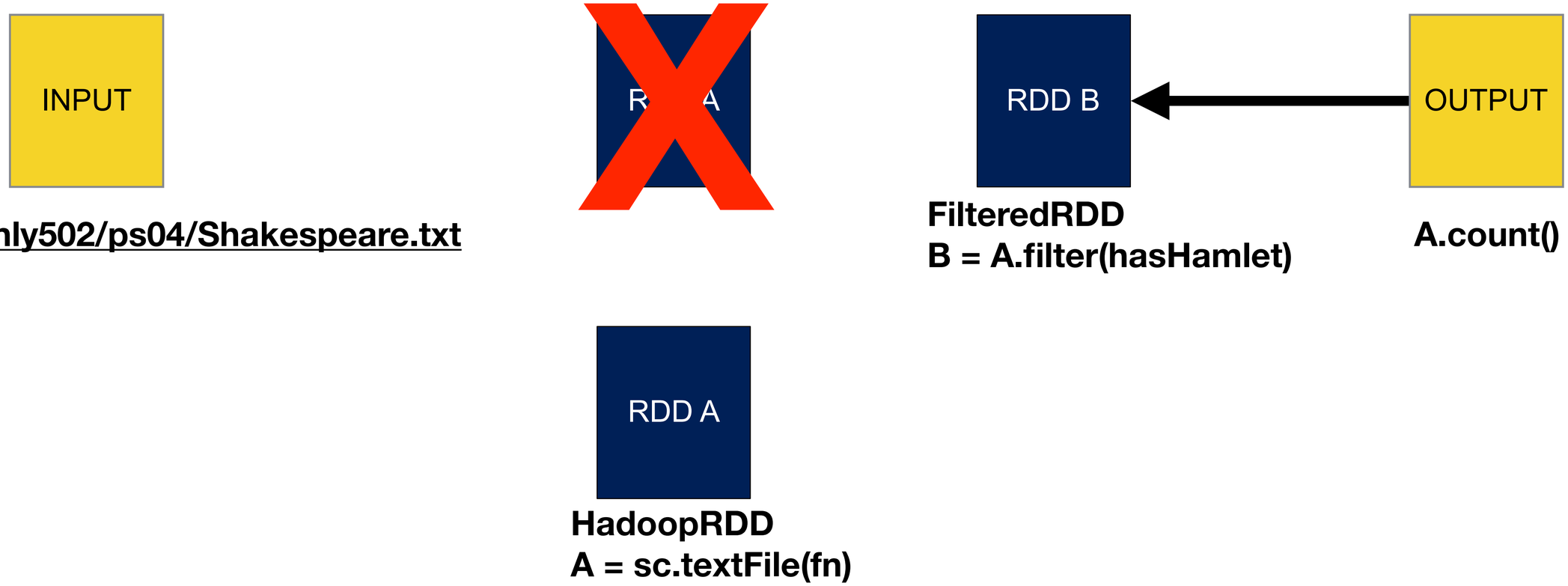


RDDs are deterministic.
If an RDD fails, it is rebuilt automatically

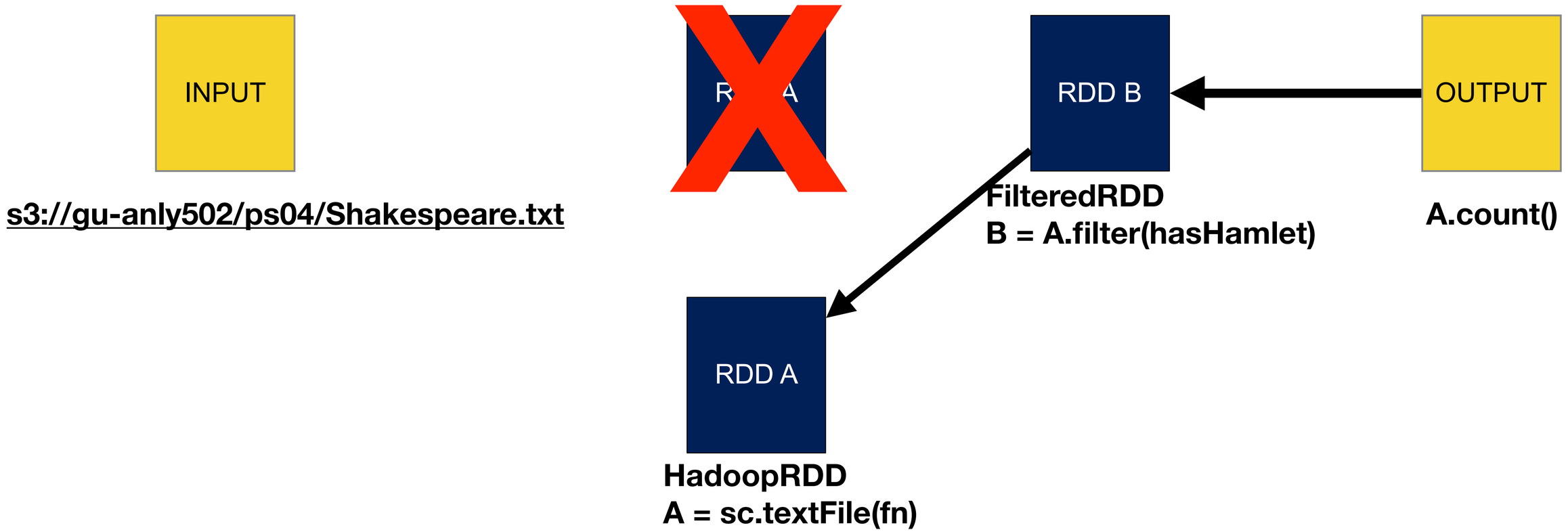


RDDs are deterministic.
If an RDD fails, it is rebuilt automatically

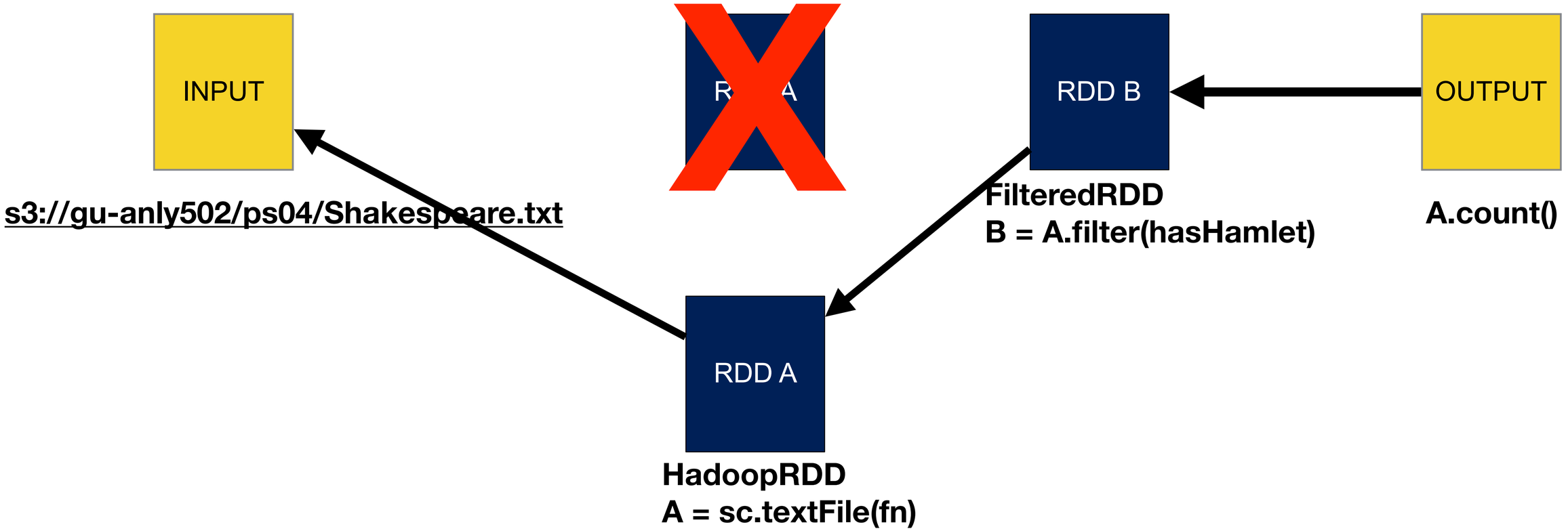
RDDs are deterministic.
If an RDD fails, it is rebuilt automatically



RDDs are deterministic.
If an RDD fails, it is rebuilt automatically



RDDs are deterministic.
If an RDD fails, it is rebuilt automatically



NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

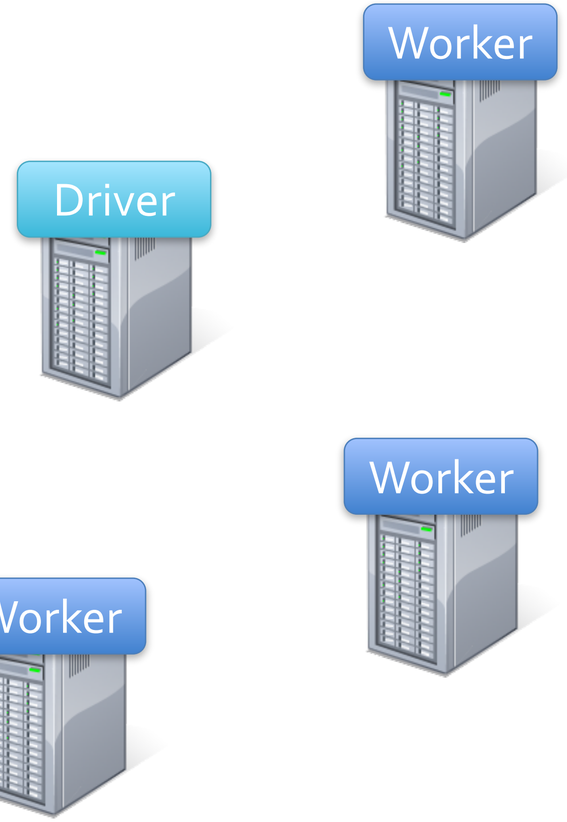
Load error messages from a log into memory, then interactively search for various patterns

NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns



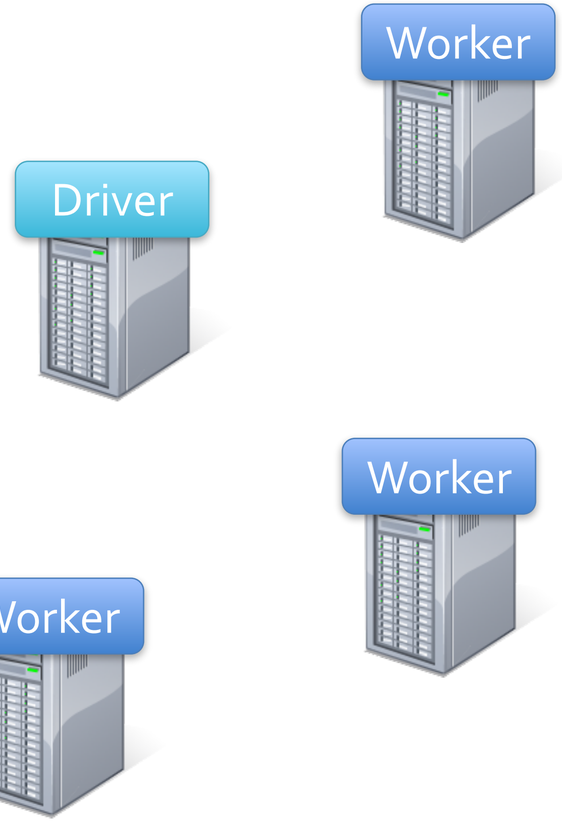
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()
```



NOTE: SCALA!

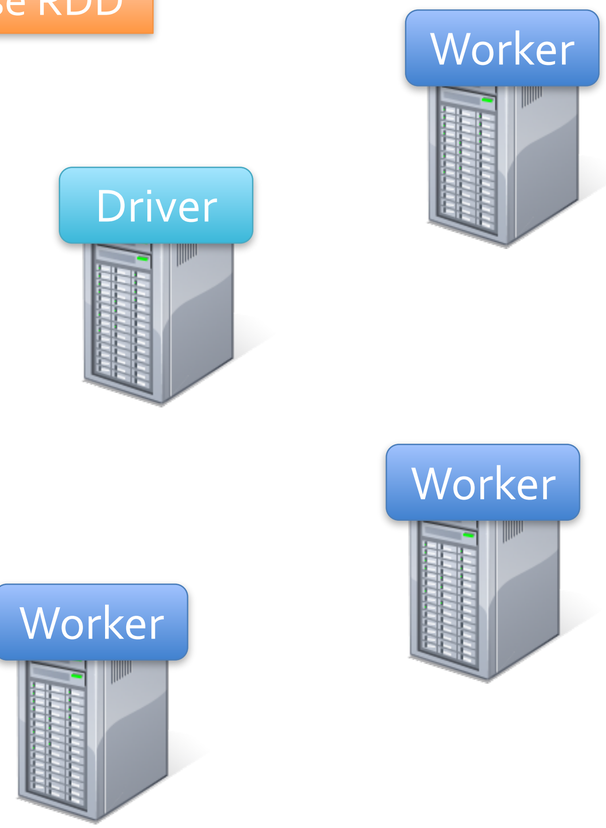
AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()
```

Base RDD



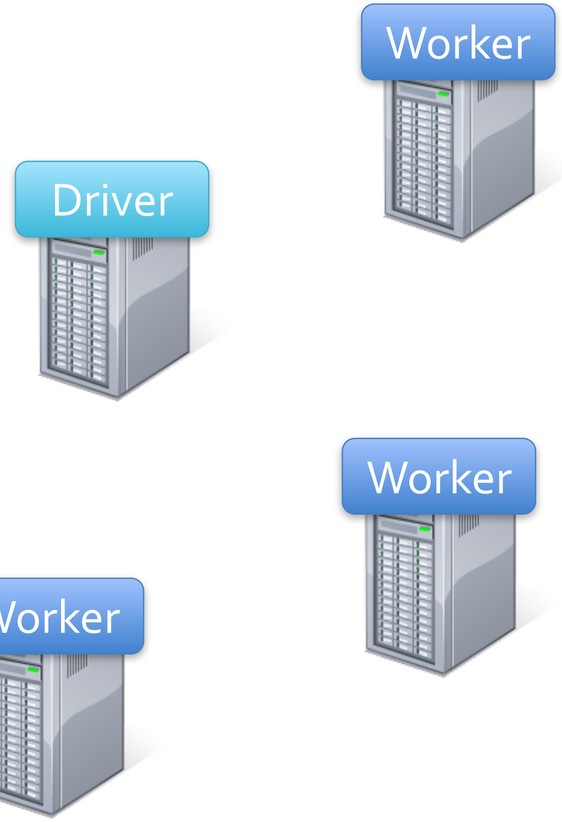
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()
```



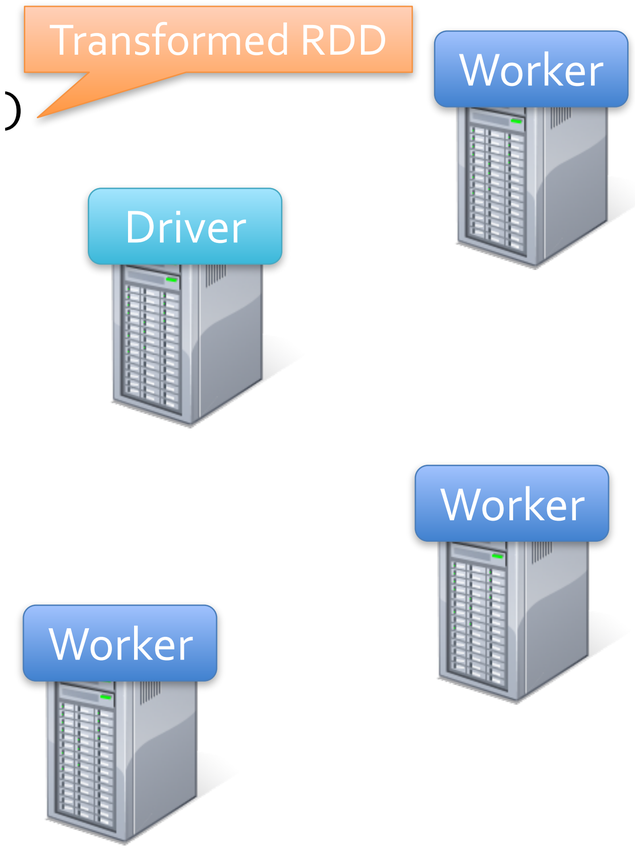
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()
```



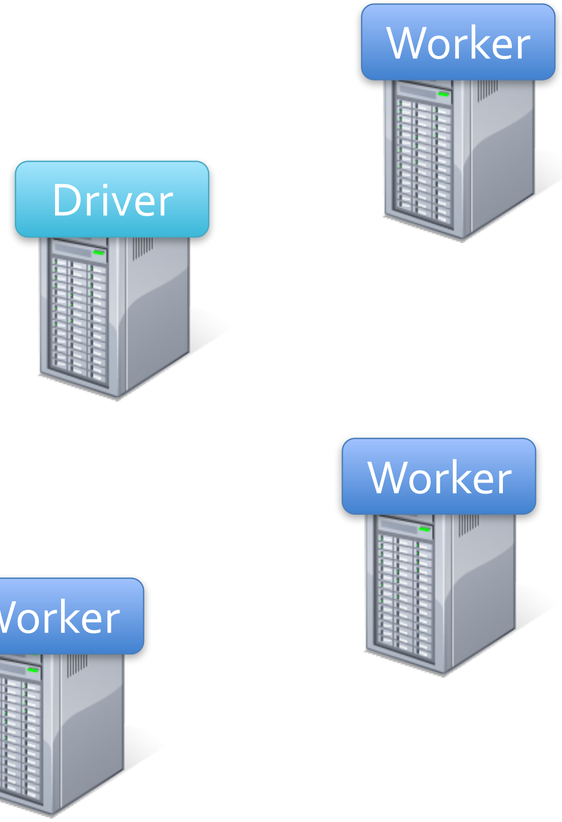
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()
```



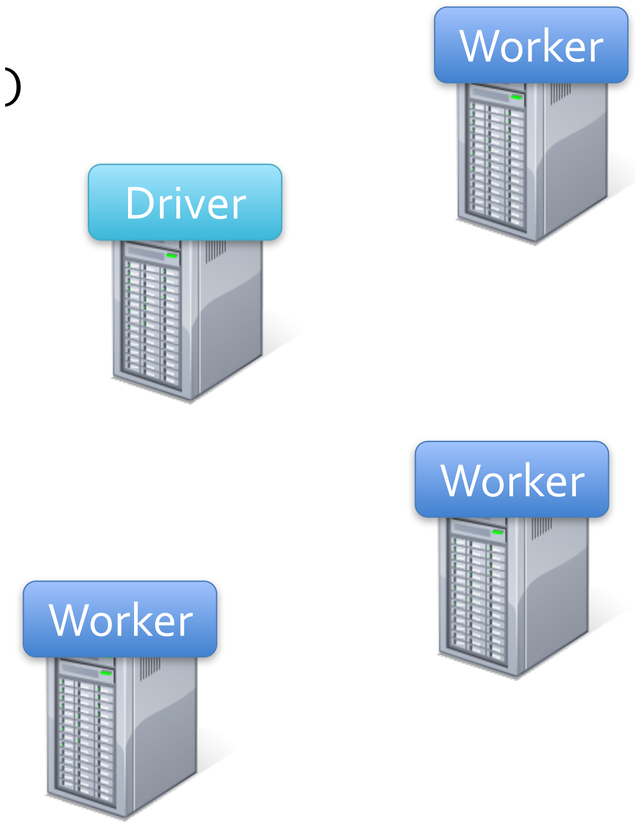
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



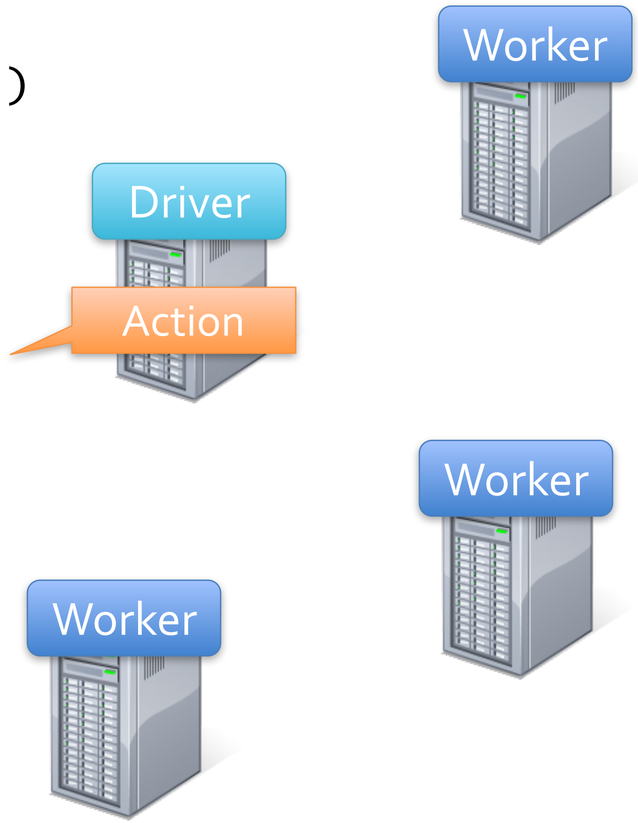
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



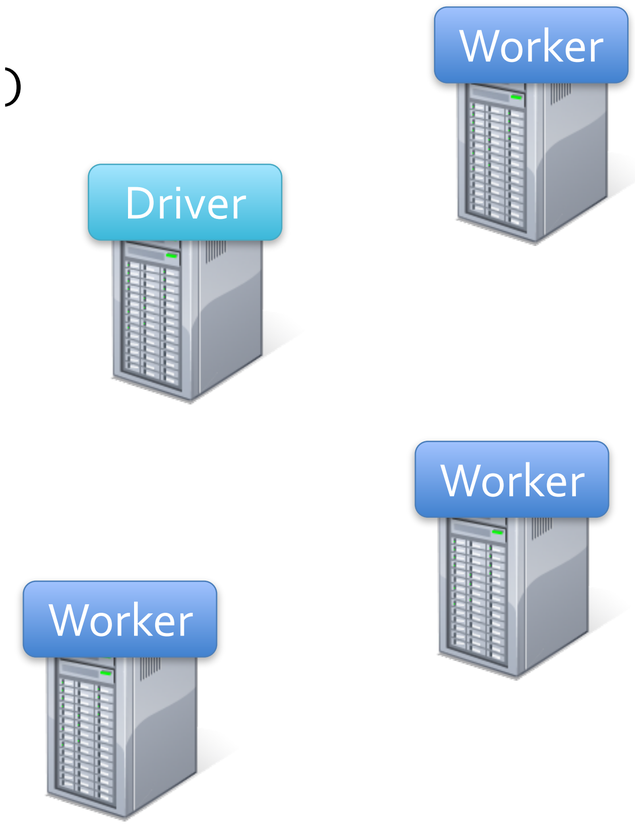
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



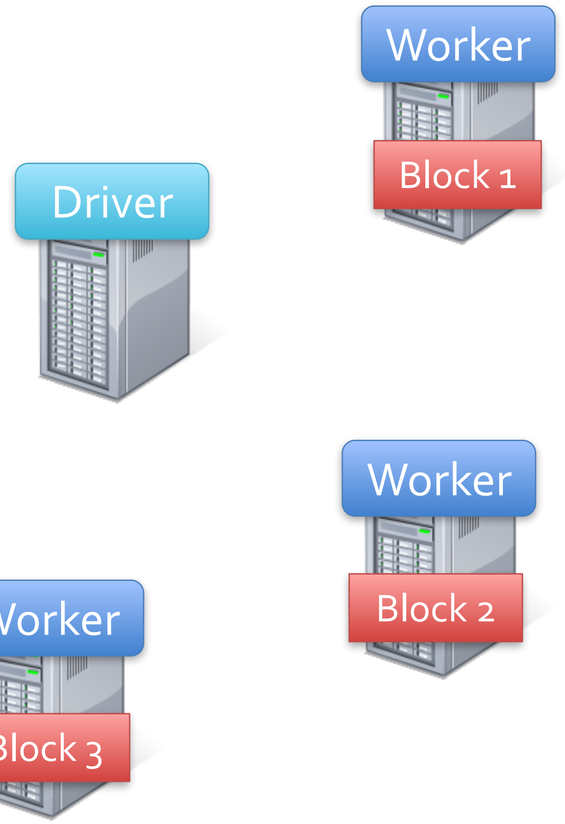
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



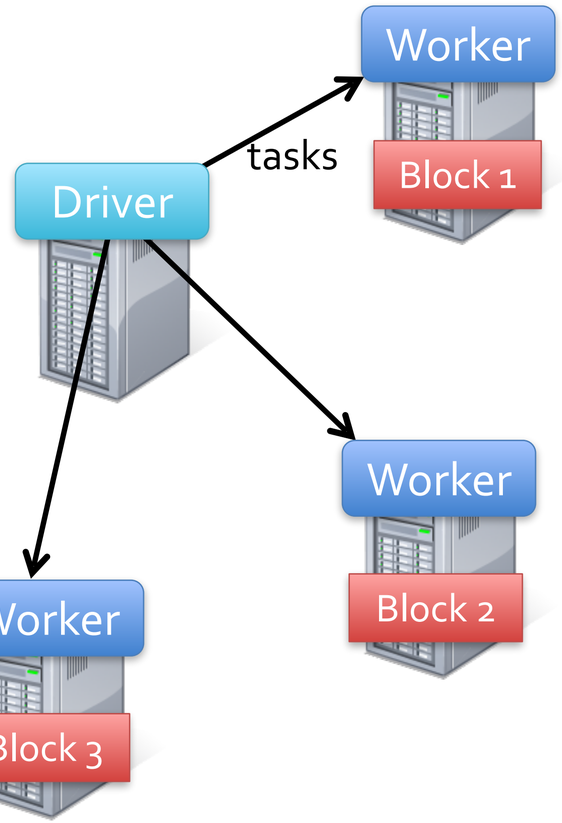
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



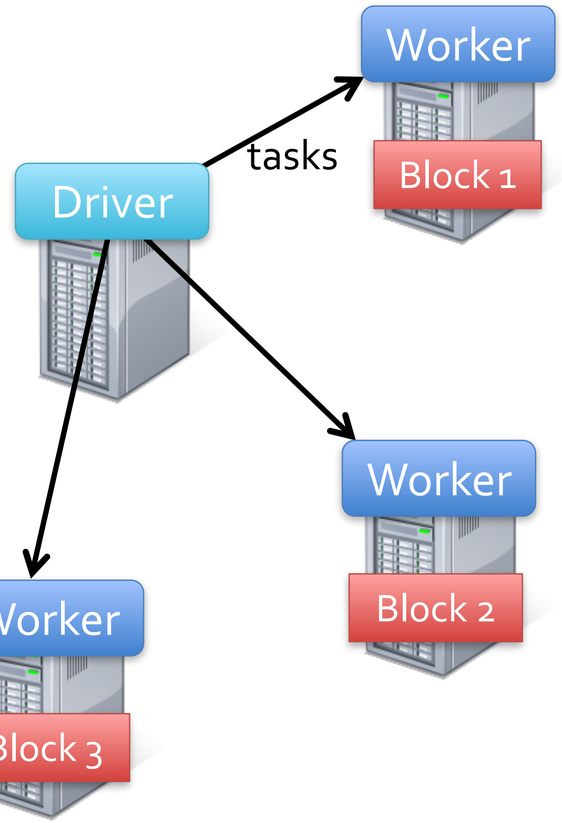
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



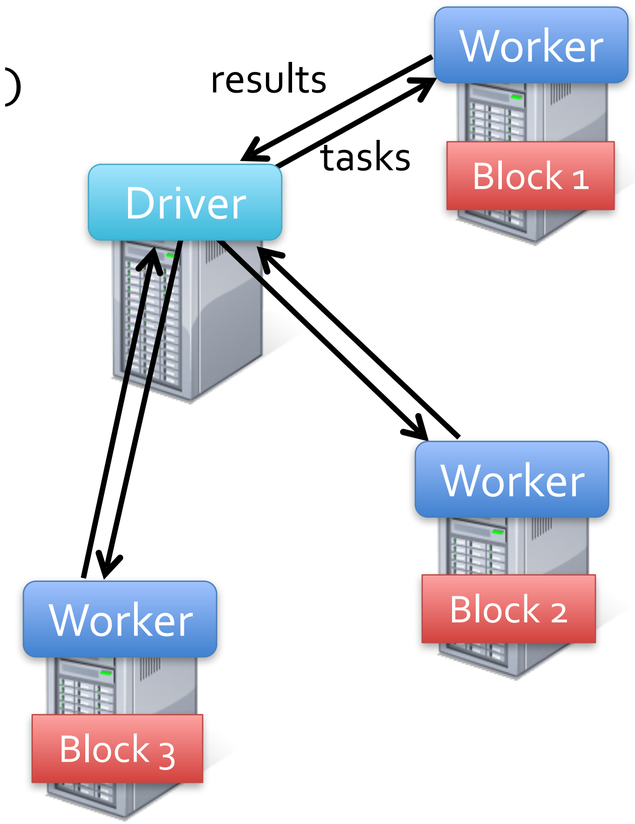
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



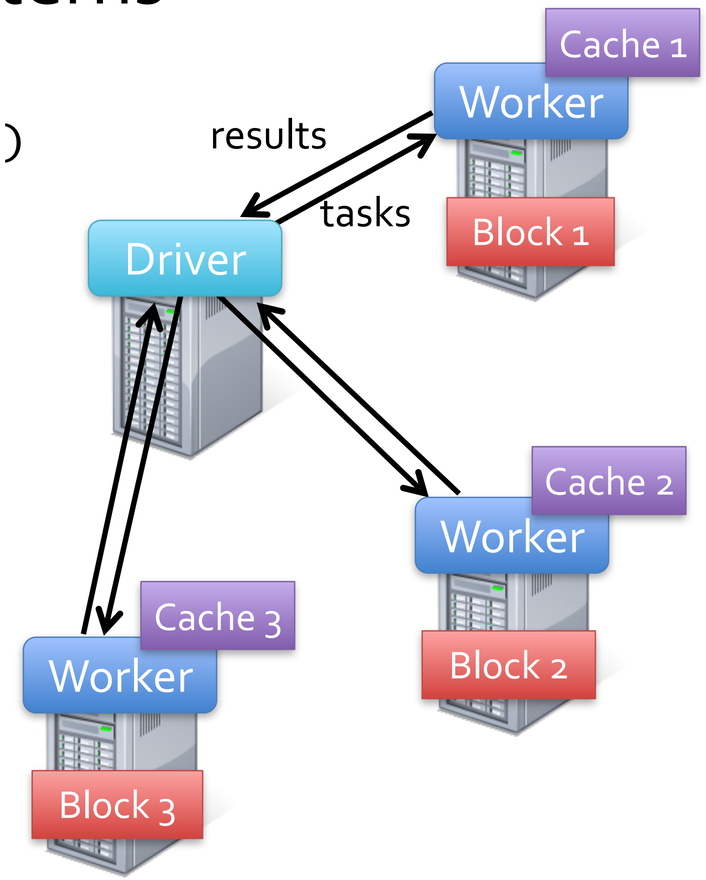
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



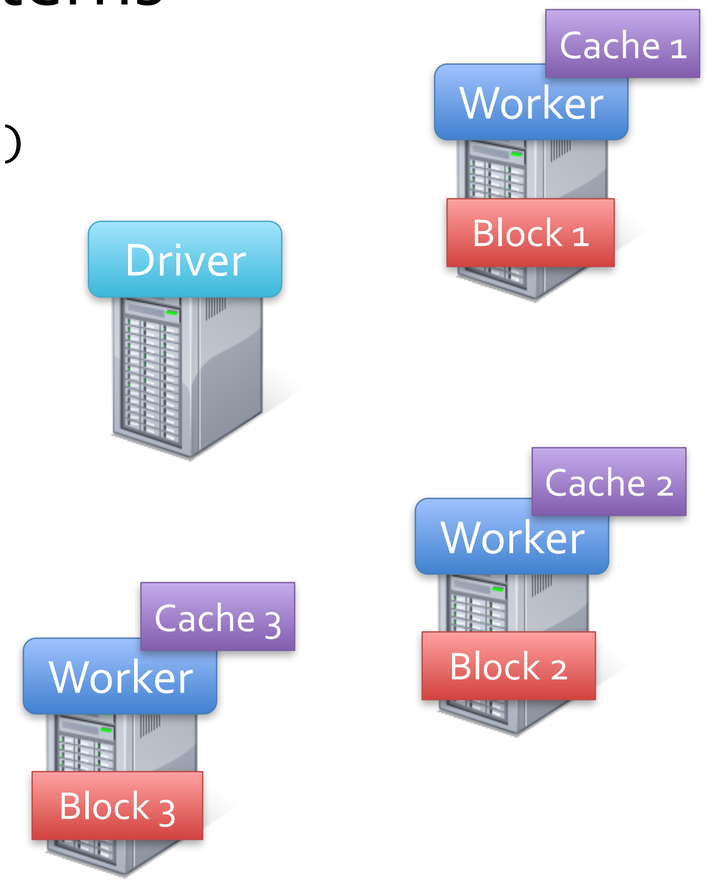
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
```



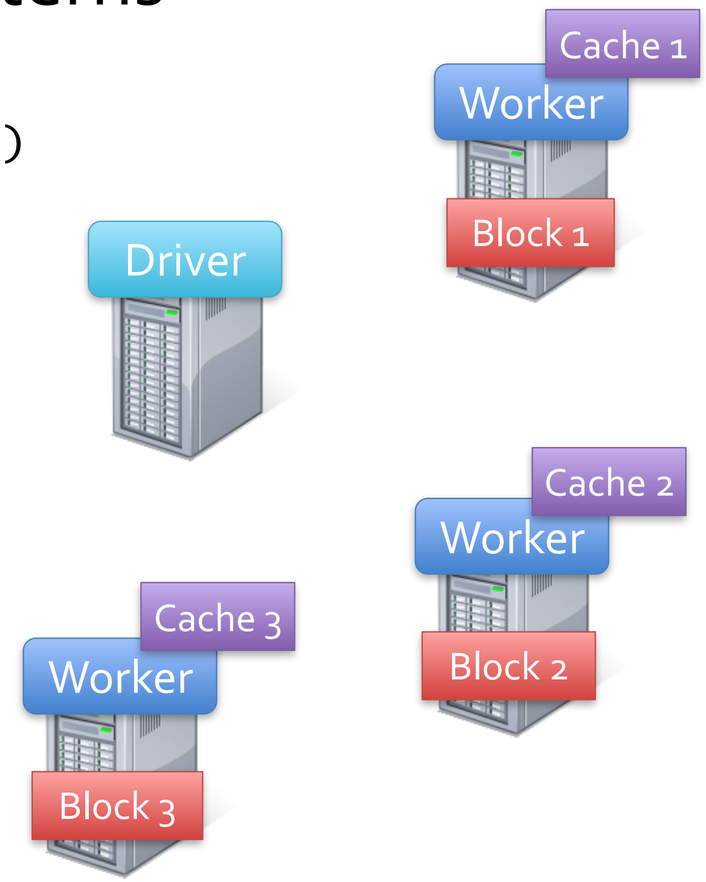
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
```



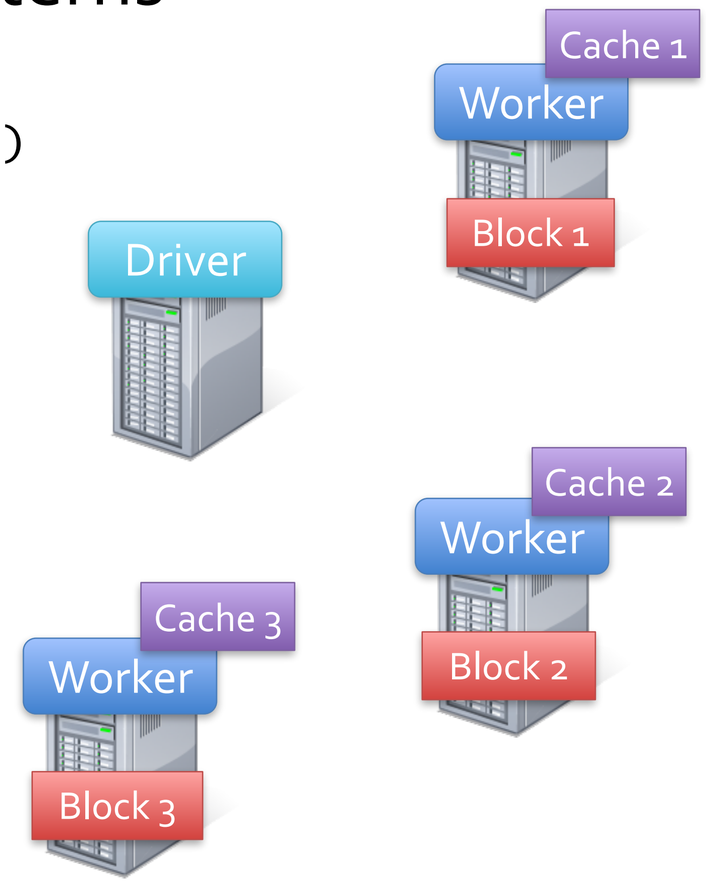
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
```



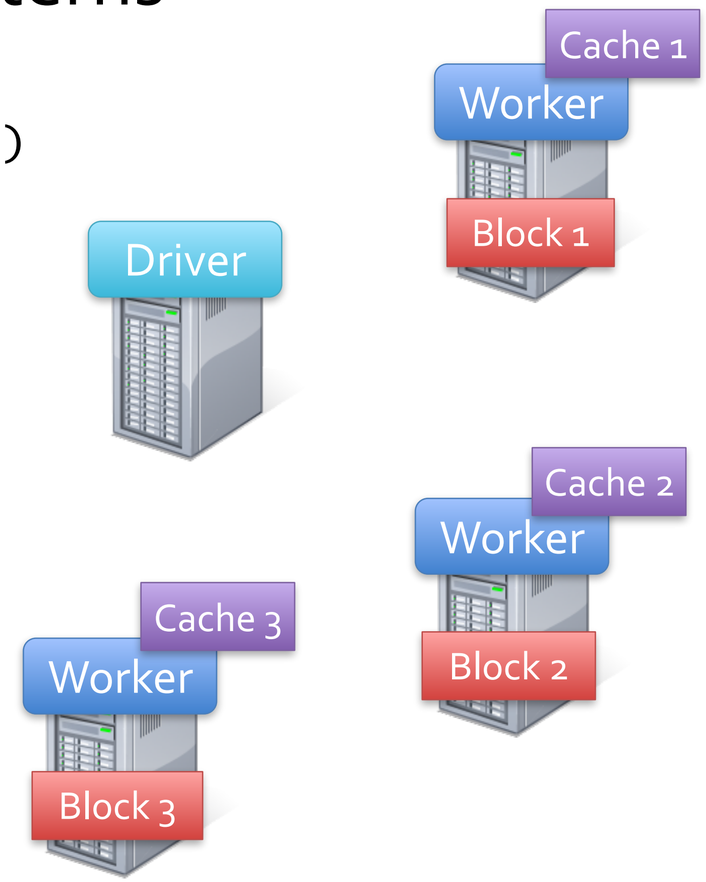
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
. . .
```



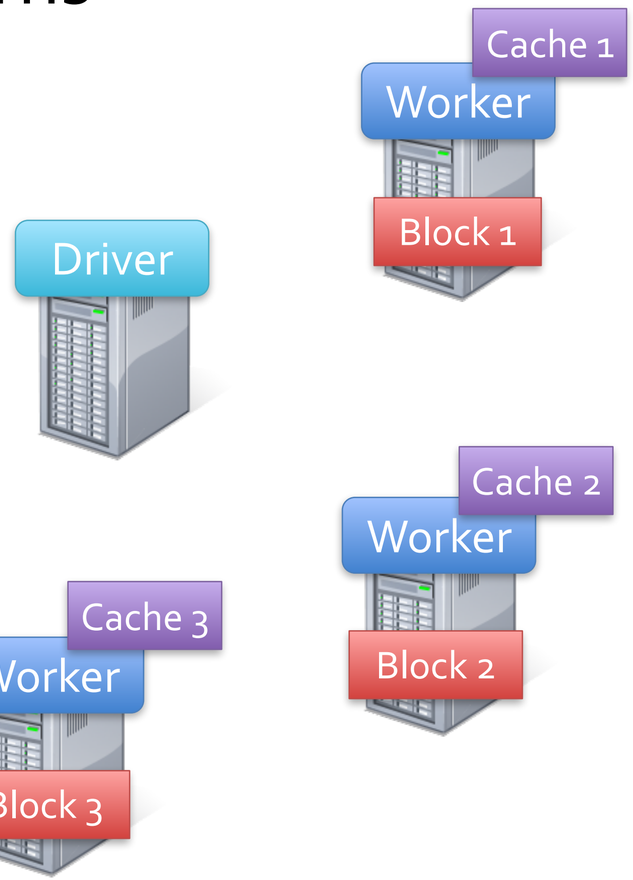
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
. . .
```



Result: full-text search of Wikipedia in <1 sec (vs 20 sec for on-disk data)

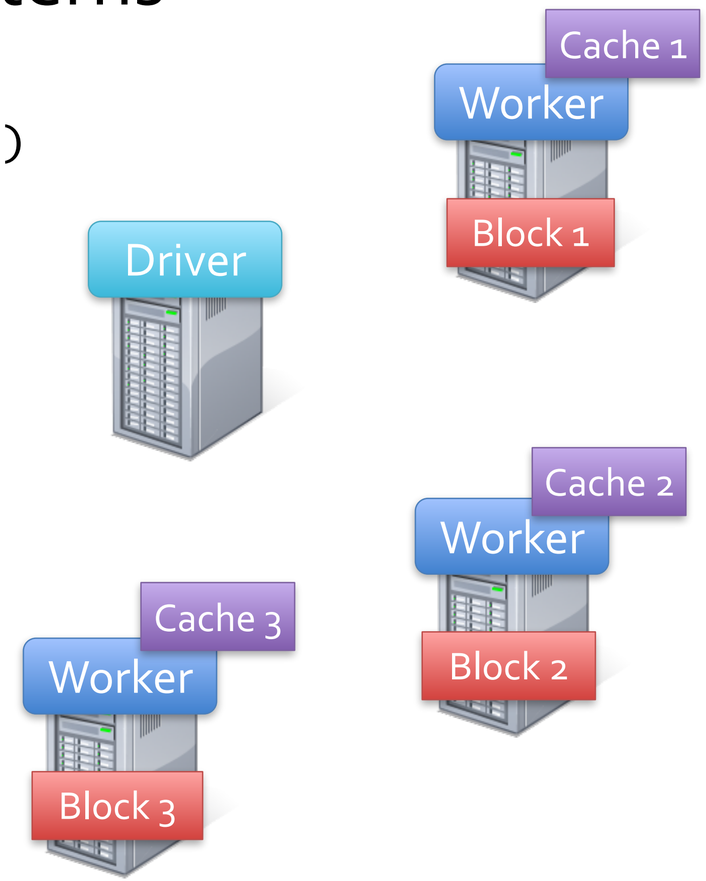
NOTE: SCALA!

AMP Lab Slide

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()


cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
. . .
```



Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)

NOTE: SCALA!

AMP Lab Slide

Example: Logistic Regression

```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

NOTE: SCALA!

AMP Lab Slide

Example: Logistic Regression

```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Load data in memory once

NOTE: SCALA!

AMP Lab Slide

Example: Logistic Regression

```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Initial parameter vector

NOTE: SCALA!

AMP Lab Slide

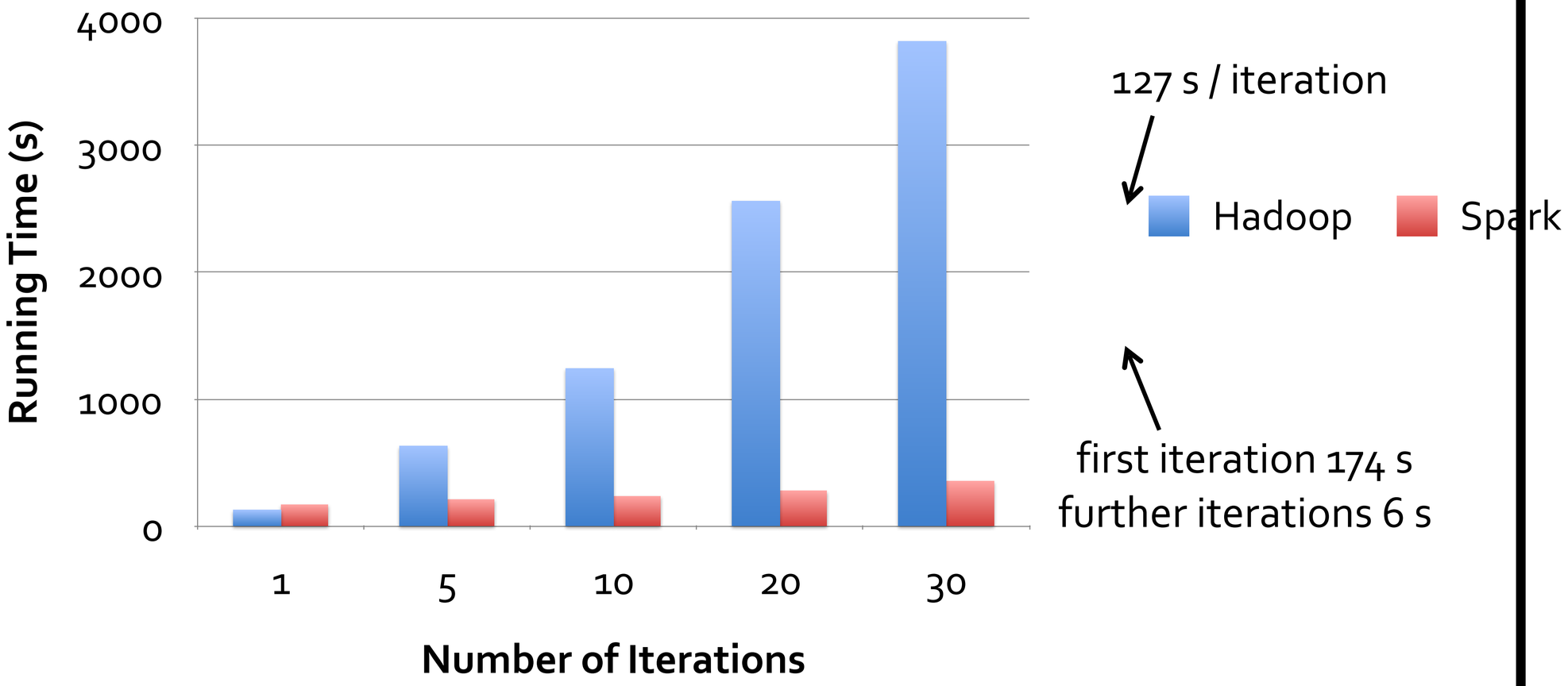
Example: Logistic Regression

```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Repeated MapReduce steps to do gradient descent

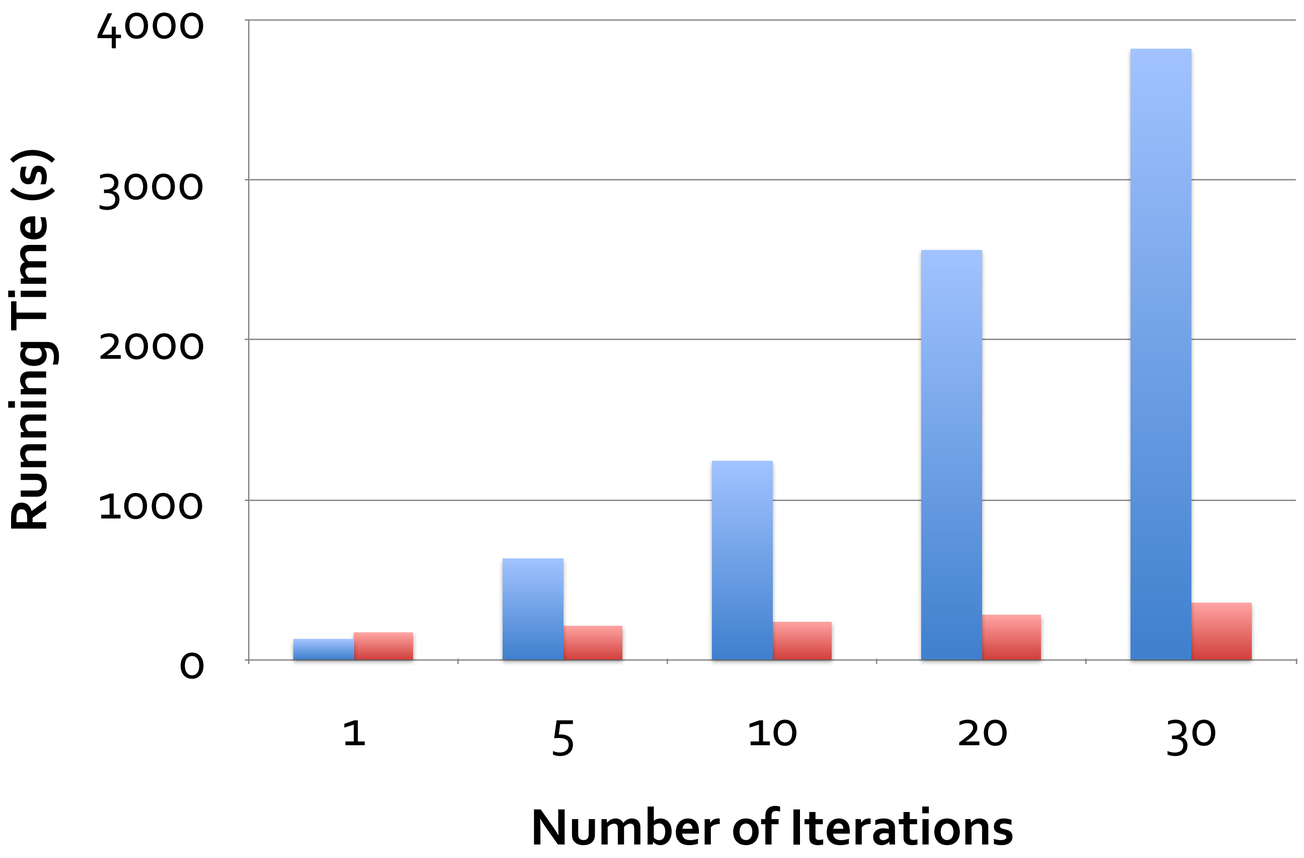
AMP Lab Slide

Logistic Regression Performance



AMP Lab Slide

Logistic Regression Performance



127 s / iteration

Hadoop Spark

first iteration 174 s
further iterations 6 s

It's faster because the data are read into memory ONCE.

Scala vs. Python
Spark was developed in Scala

Expect to see code like this:

- lines = spark.textFile(“hdfs://...”)
- errors = lines.filter(_.startsWith(“ERROR”))
- messages = errors.map(_.split(‘\t’)(2))
- cachedMsgs = messages.cache()

Translate it like this:

- lines = sc.textFile(“hdfs://...”)
- errors = lines.filter(lambda a:a.startswith(ERROR"))
- messages = errors.map(lambda a:a.split("\t", 1)
- cachedMsgs = messages.cache()

Good news: Python and scala look a lot alike.

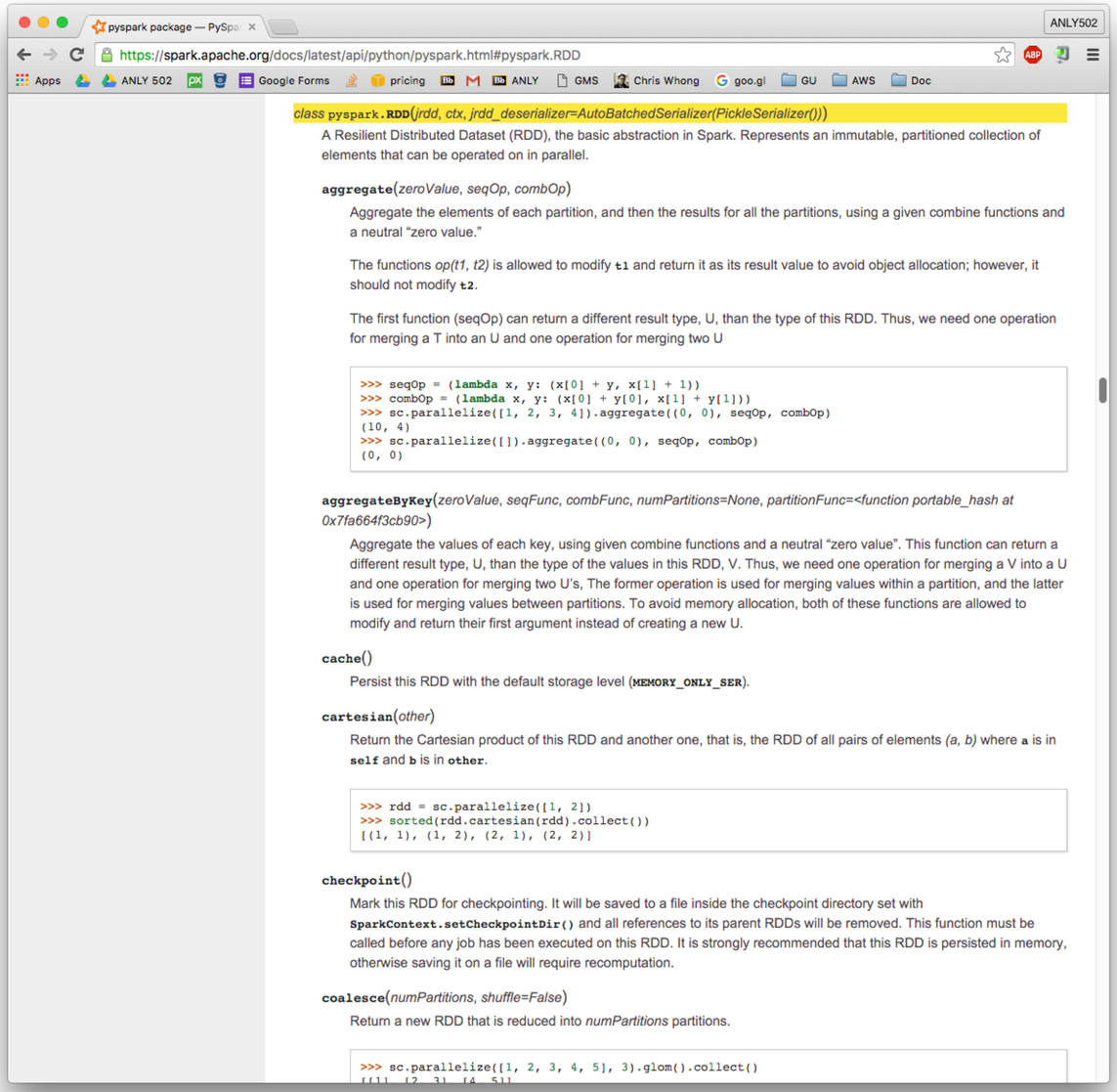
pyspark.RDD — basic class for RDD

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

About 100 different methods

- map, reduce, reduceByKey
- filter
- count
- cogroup, groupBy
- partitionBy
- join, leftOuterJoin, rightOuterJoin, cross
- sample
- save
- pipe

More...



About Spark

Who is using Spark?

Spark history and community

2009 — Developed at UC Berkeley AMPLab

2010 — First open source release

2013 — Donated to Apache Foundation

2014 — Spark becomes a "Top-Level Apache Project"

2015 — Spark has more than 1000 contributors.

	June 2014	June 2015
Total contributors	255	730
Contributors/month	75	135
Lines of code	175,000	400,000

Large-Scale Usage

Largest cluster: 8000 nodes Tencent 腾讯

Largest single job: 1 petabyte 阿里巴巴 Alibaba.com databricks

Top streaming intake: 1 TB/hour HHMI janelia farm research campus

2014 on-disk sort record

Spark ecosystem:

Spark Summit 2015

Users

1000+ companies

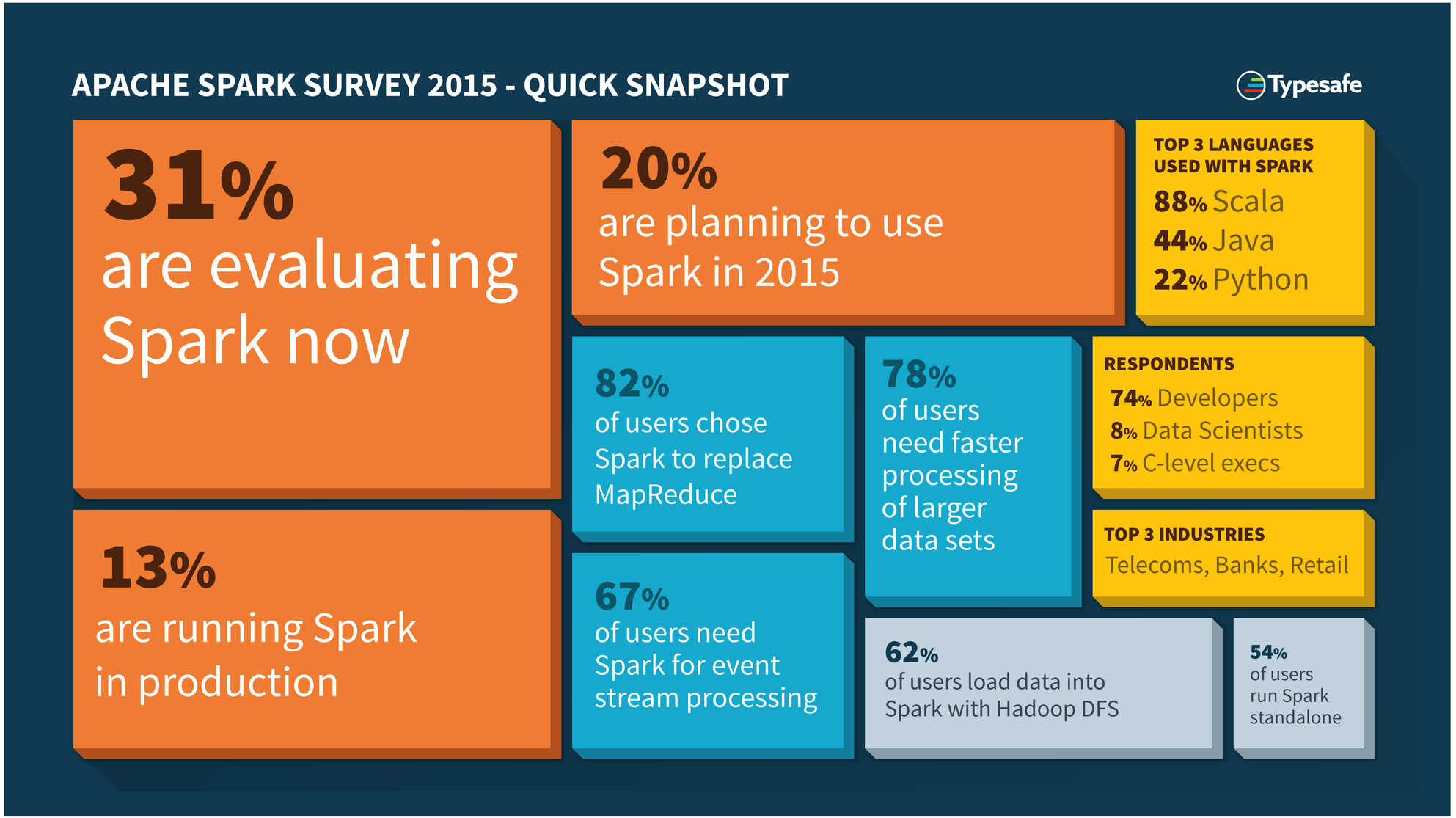
. . .

Distributors + Apps

50+ companies

. . .

Typsesafe 2015 Spark Survey — 2136 respondents throughout industry. Main finding: Lots of companies are *thinking about Spark**

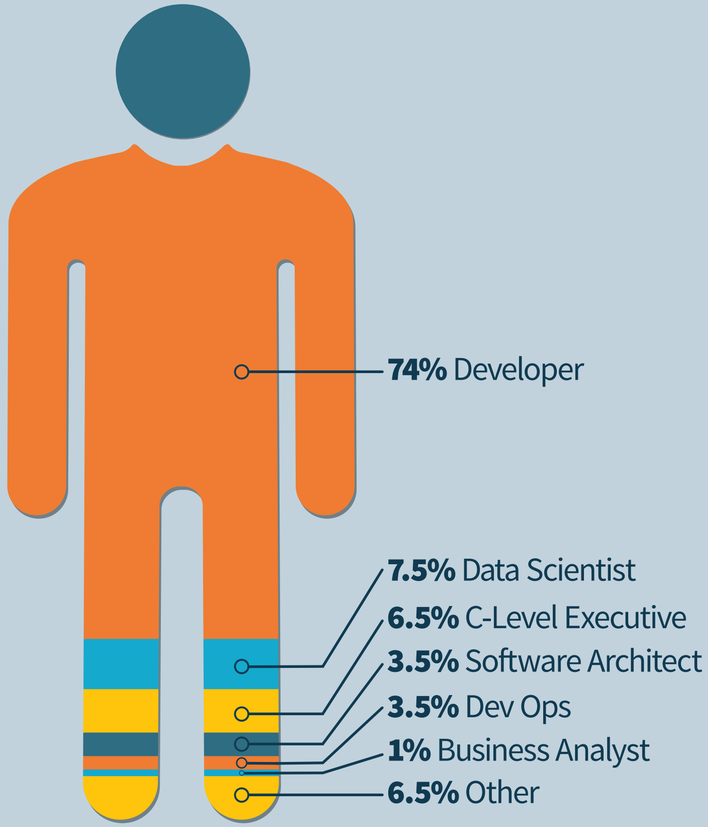


https://dzone.com/articles/apache-spark-survey-typesafe-0

*Which is why we also taught MapReduce & Pig

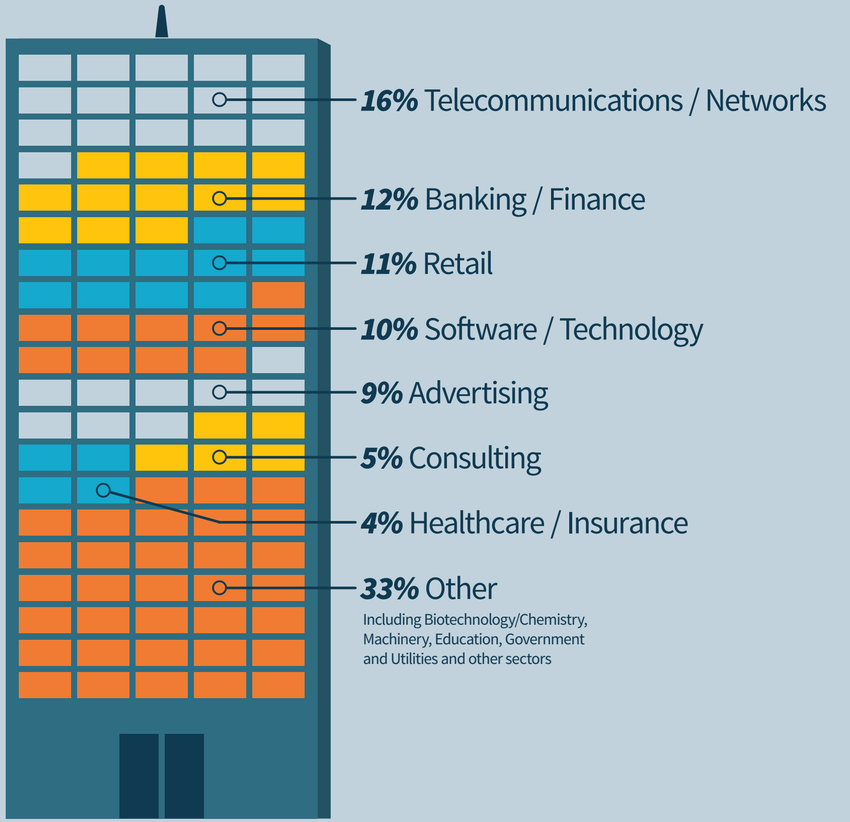
Typesafe respondents…

JOB TYPE/ROLE



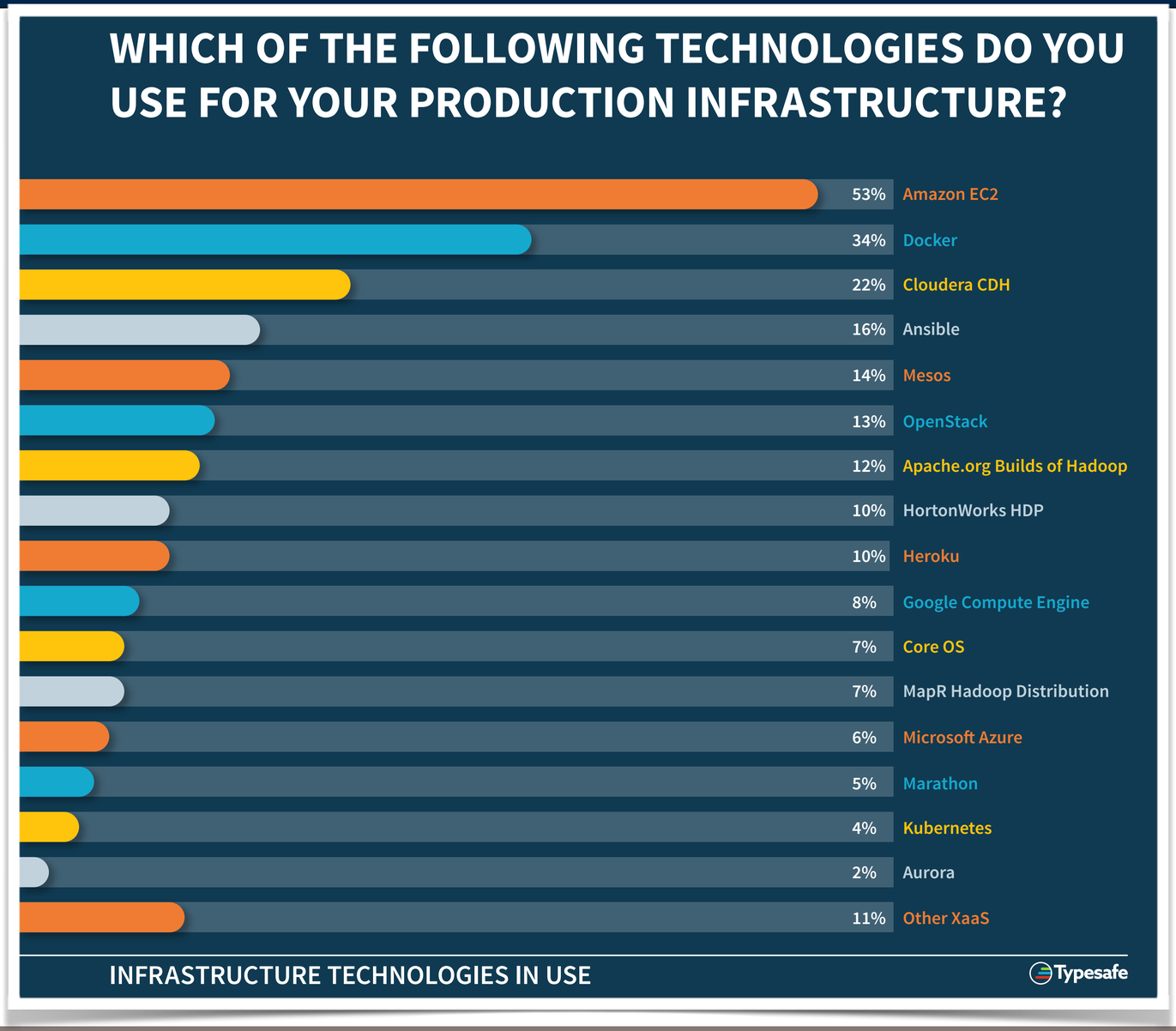
Typesafe

INDUSTRY FOCUS

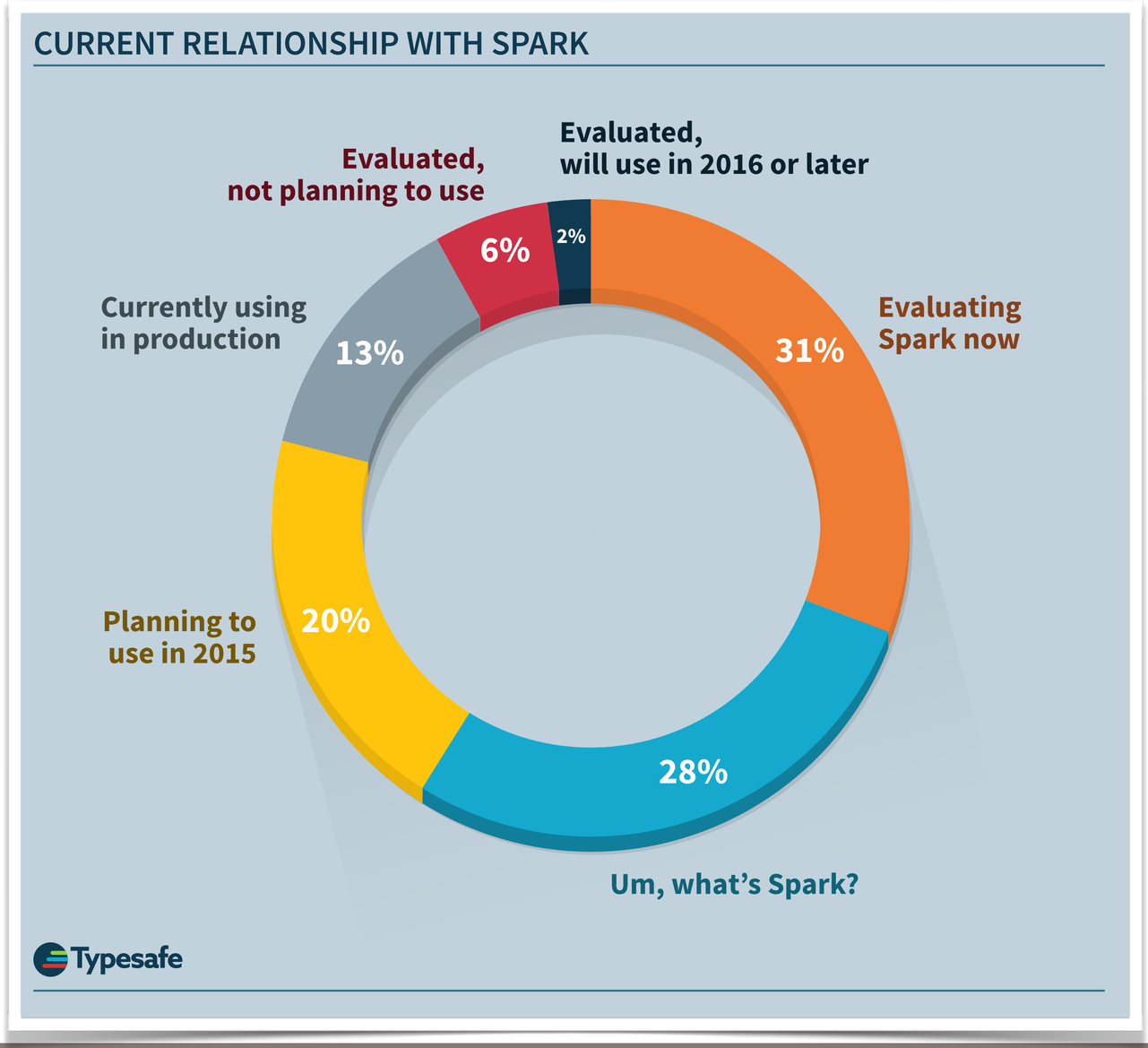


Typesafe

More than 1/2 are using Amazon EC2

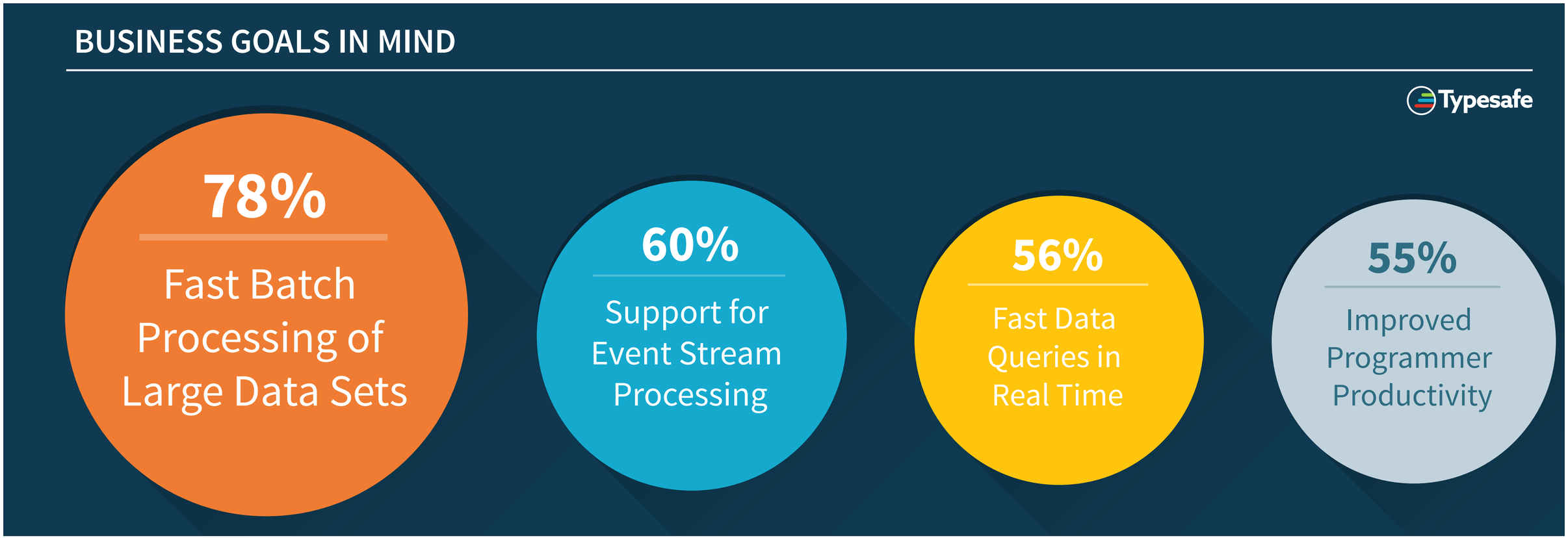


But only 13% are actually using Spark in production (in 2014)

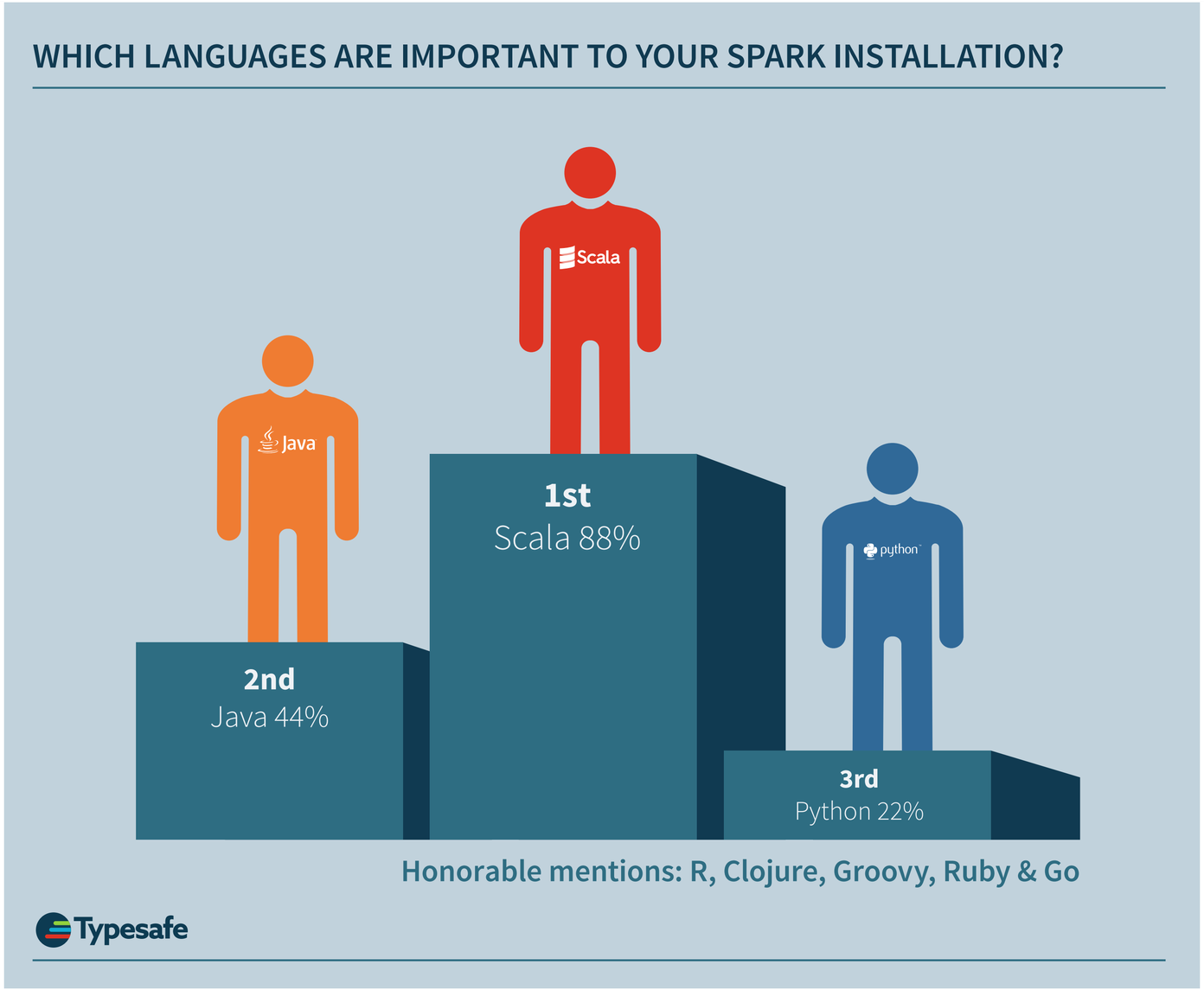


Most want to use Spark for faster batch processing.

"What problems are you trying to solve with Spark that other tools don't solve?"



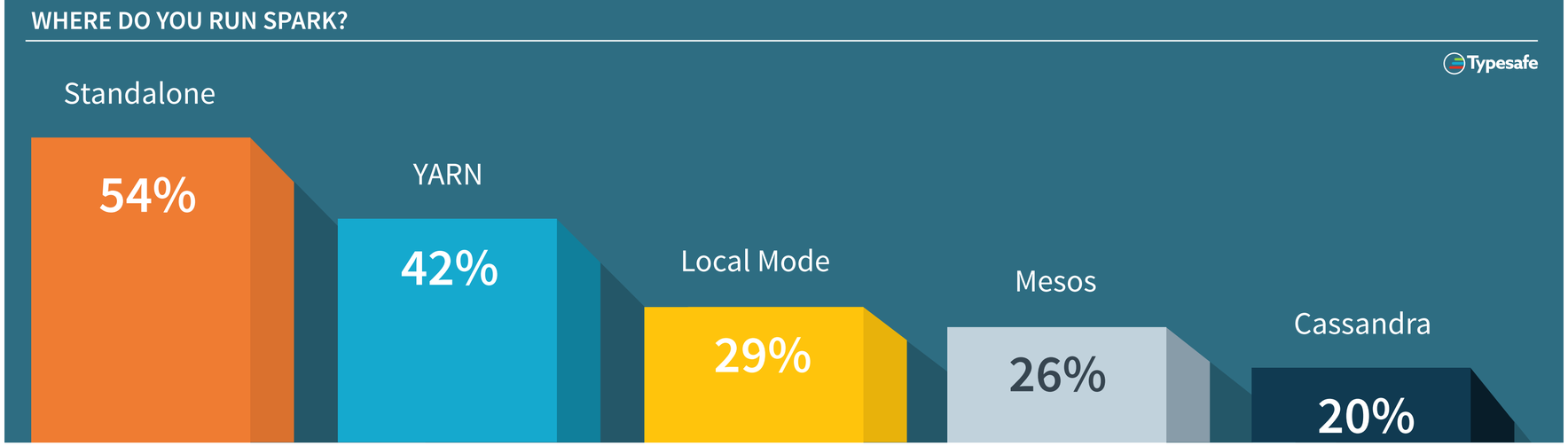
Python: Spark's least popular language.



We are running Spark on top of YARN — a popular configuration.

WHERE ARE YOU RUNNING SPARK CURRENTLY?

Standalone (54%) and **Local mode** (29%) installations of Spark seem logical for early users with different testing purposes, and one can always add to a cluster later. Otherwise, **YARN** (42%), aka MapReduce 2, and **Mesos** (26%) are the general go-to choices for integrating and running Spark with current systems. **Cassandra** (20%) is another Apache project that not only integrates well with Spark's event streaming power, but shares a similar vision of supporting highly responsive, resilient, elastic systems. Also mentioned by about 3% of respondents is Amazon Elastic MapReduce.



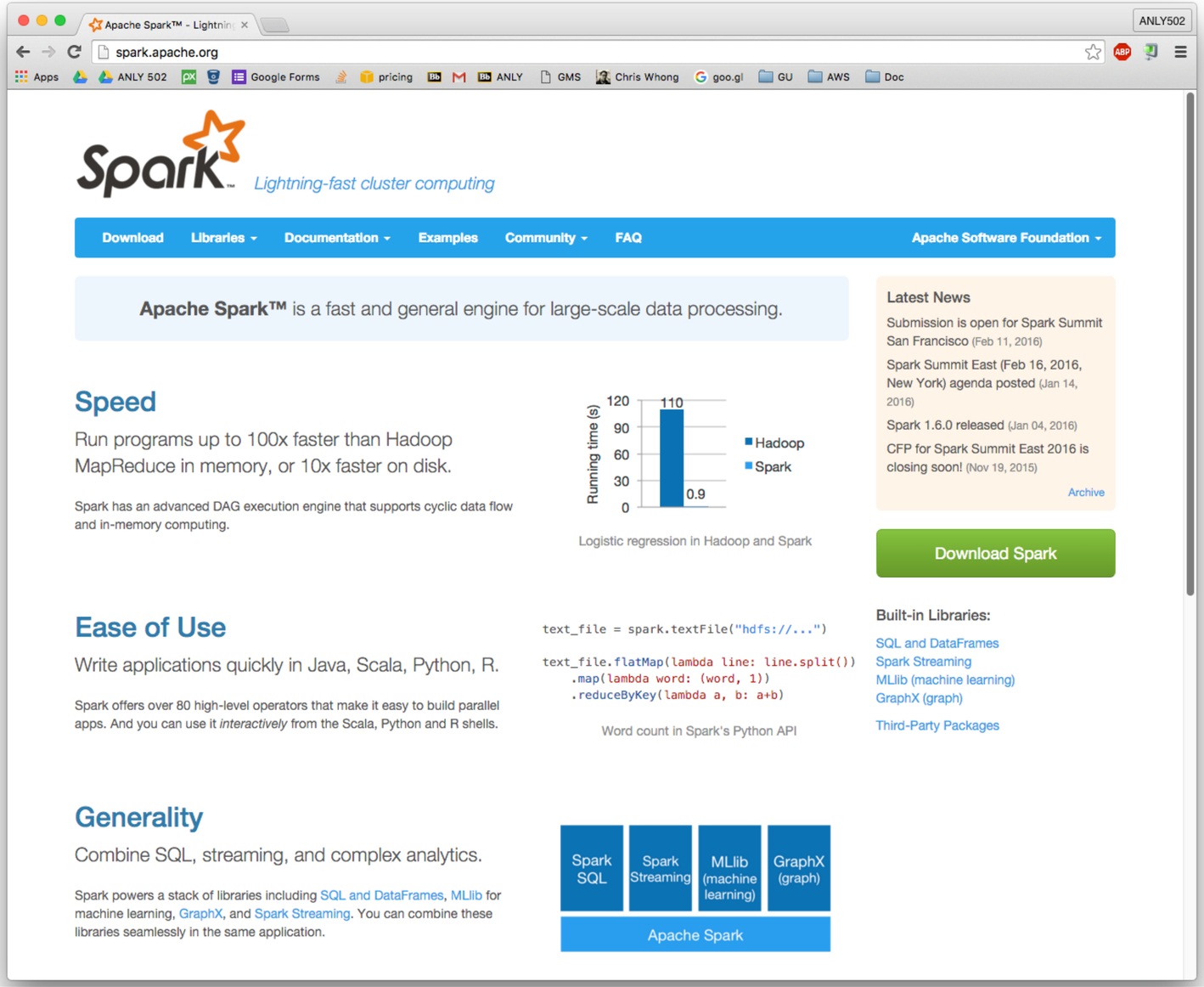
S3 is not that popular for loading data into Spark

Please log in to EMR and start a 1-node cluster!

Be sure to use s3://gu-anly502/bootstrap.sh

Spark Resources

http://spark.apache.org/ — Your primary resource



spark.apache.org

Documentation

- https://spark.apache.org/docs/latest/

Examples:

- https://spark.apache.org/examples.html

API:

- https://spark.apache.org/docs/latest/api/python/index.html

Word Count

In this example, we use a few transformations to build a dataset of (String, Int) pairs called `counts` and then save it to a file.

Python | Scala | Java

```python
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://...")
```

Python | Scala | Java

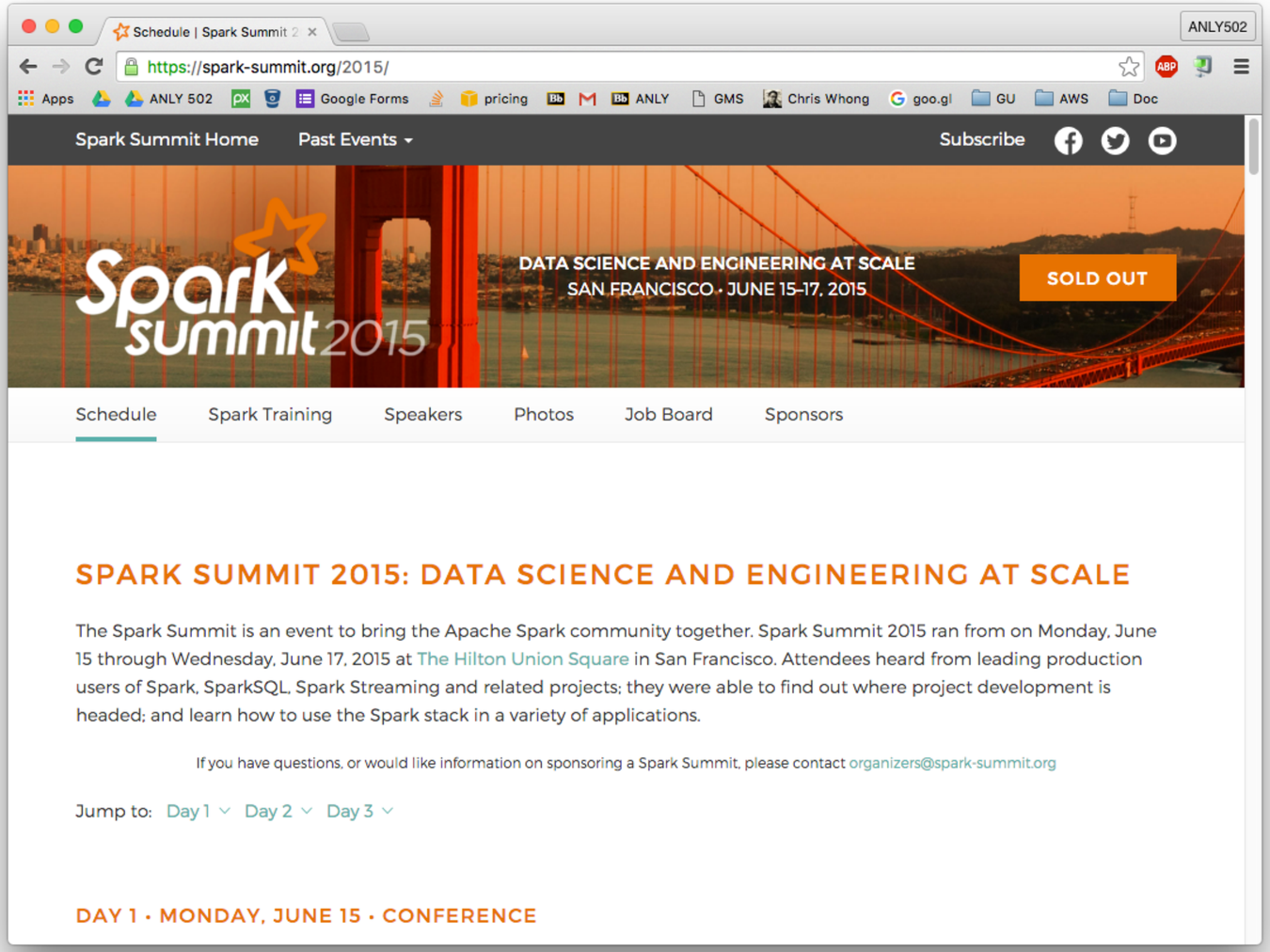
```scala
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

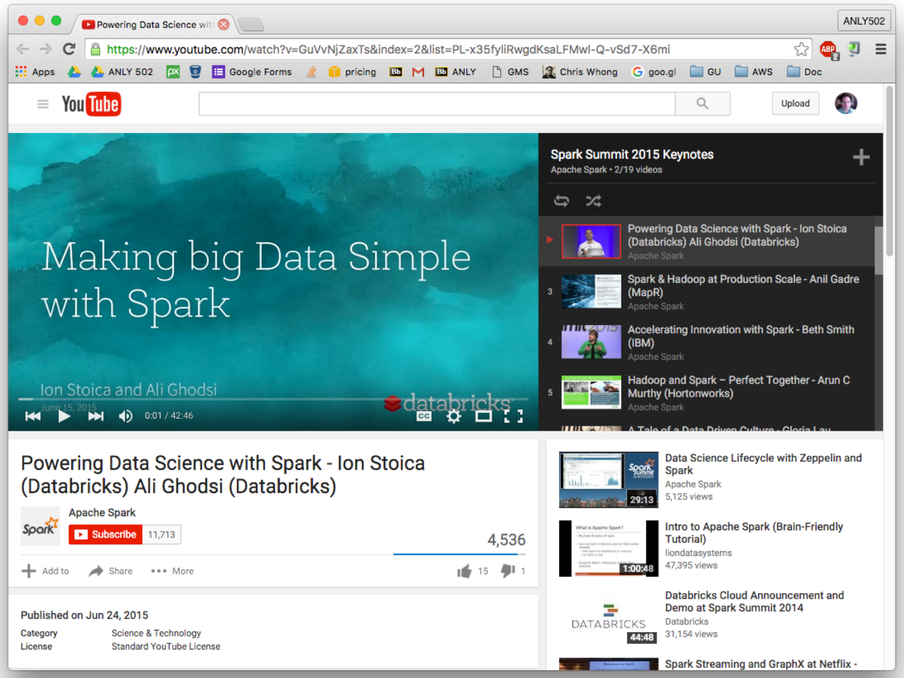
Python | Scala | Java

```java
JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaRDD<String> words = textFile.flatMap(new FlatMapFunction<String, String>() {
  public Iterable<String> call(String s) { return Arrays.asList(s.split(" ")); }
});
JavaPairRDD<String, Integer> pairs = words.mapToPair(new PairFunction<String, String, Int
eger>() {
  public Tuple2<String, Integer> call(String s) { return new Tuple2<String, Integer>(s,
1); }
});
JavaPairRDD<String, Integer> counts = pairs.reduceByKey(new Function2<Integer, Integer, I
nteger>() {
  public Integer call(Integer a, Integer b) { return a + b; }
});
counts.saveAsTextFile("hdfs://...");
```

Spark Summit — Annual conference with information about Spark https://spark-summit.org/2015/

Slides, code & videos — Great source for what's new.





You can do a lot with a relatively small cluster.

Use of Spark MLlib for Predicting Offlining of Digital Media

Cluster Configurations

- Development
  - -1 Linux System with 16 cores and 16 Gigs RAM
  - -3 Mac OS Systems each with 8 cores and 16 Gigs RAM
  - -Mesos cluster manager
- Production
  - -3 Linux systems with 32 cores and 64 Gigs of RAM
  - -Mesos master and slaves running in Docker containers
  - -

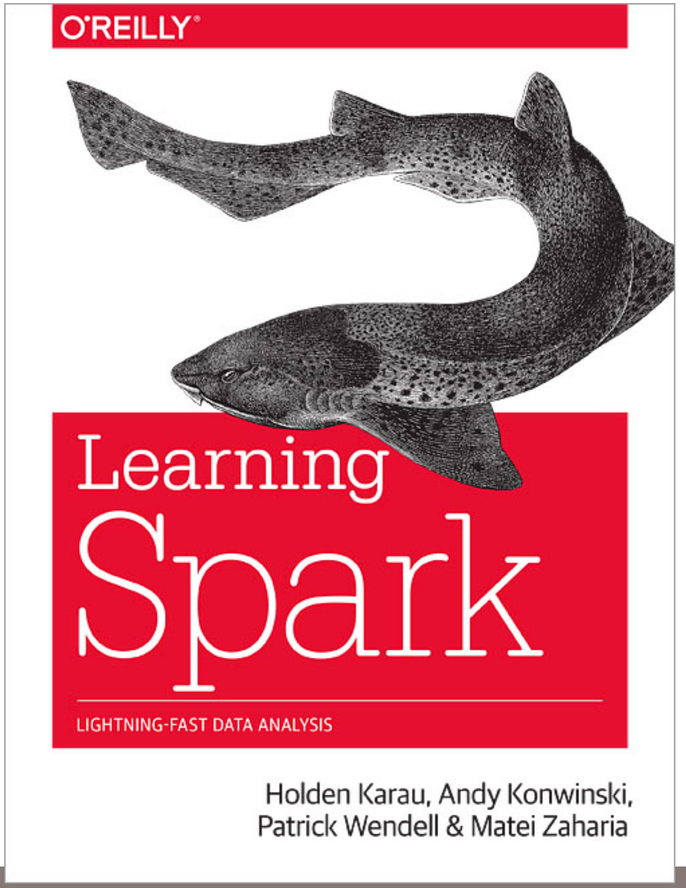
3x32 = 96 cores
3x64 = 192 GiB RAM

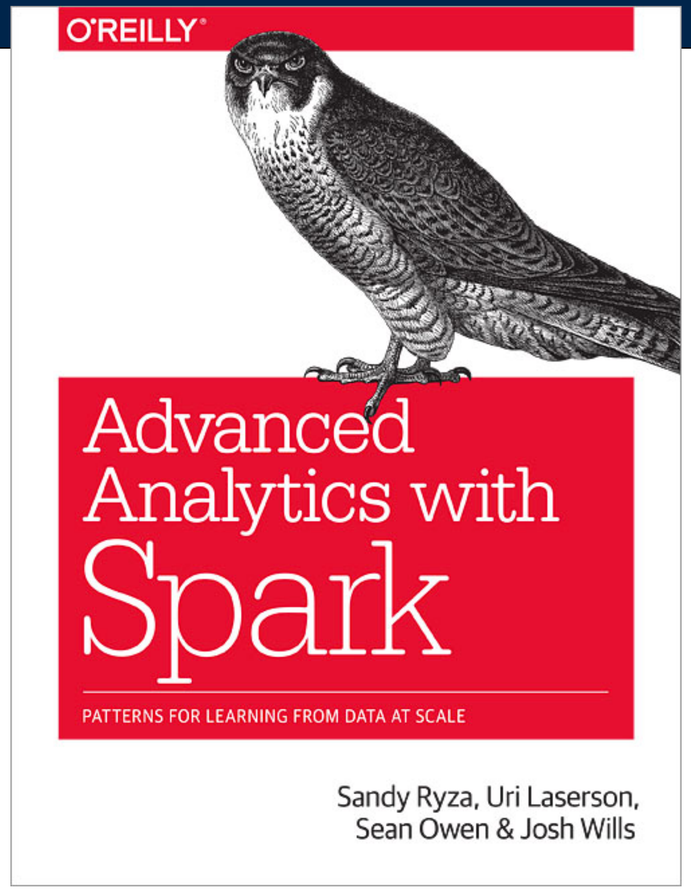
Christopher Burdorf, NBC Universal

O'Reilly books…

Learning Spark (O'Reilly 2015)

- https://github.com/databricks/learning-spark
- Scala, Java, & Python
- Concentrates on mechanisms, rather than analytical goals.

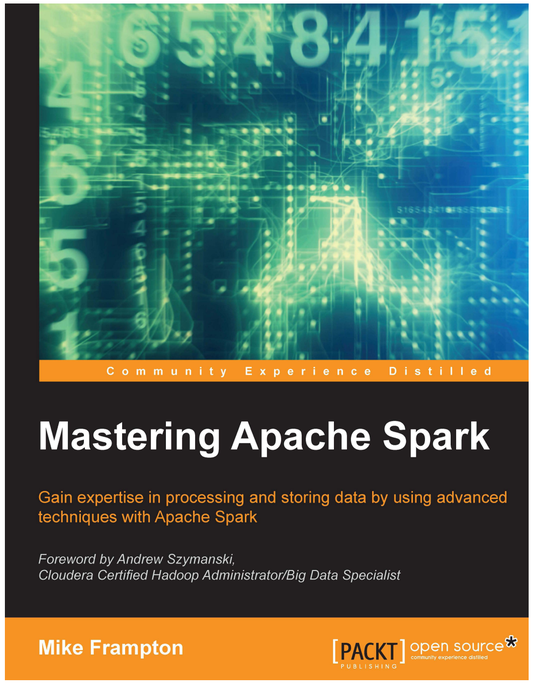


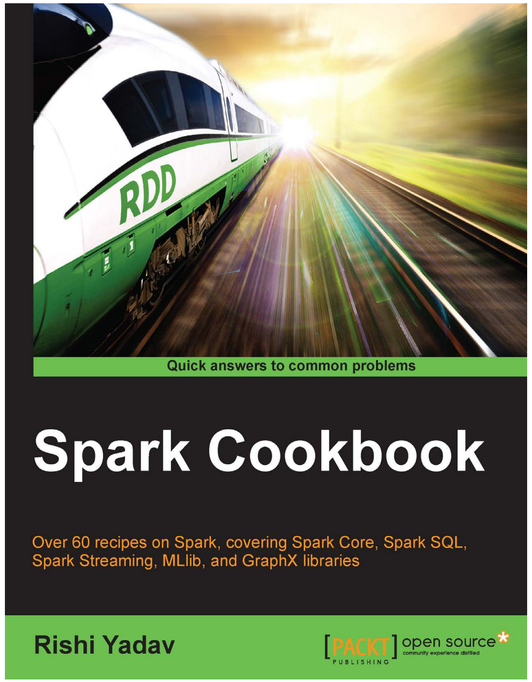


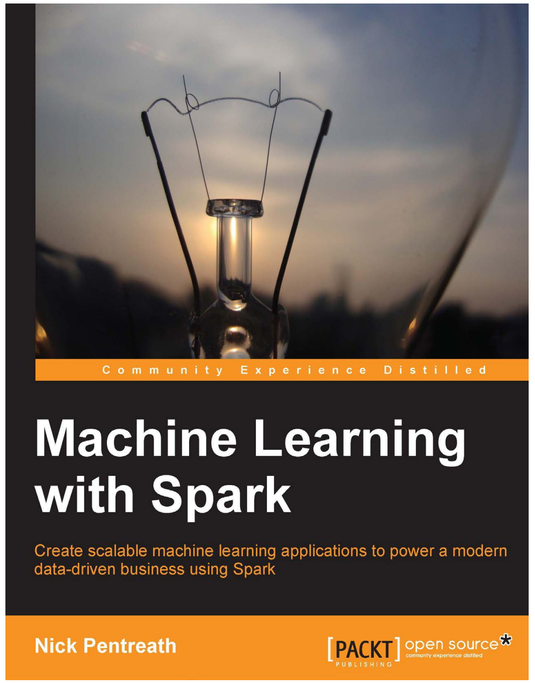
Advanced Analytics with Spark

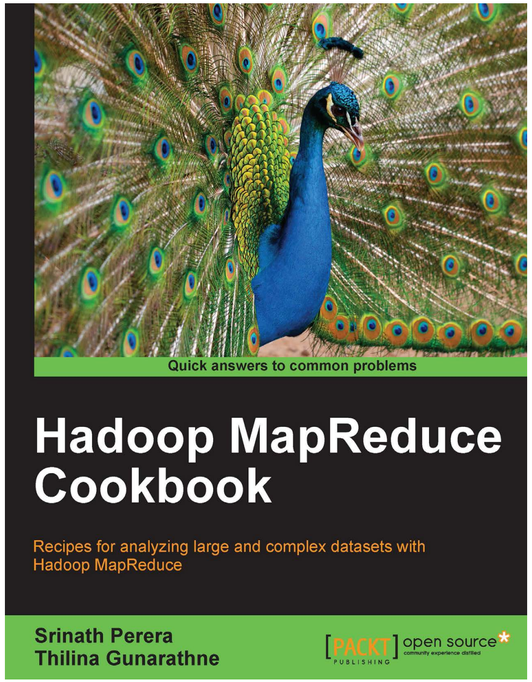
- Several worked examples
- Scala only
- https://github.com/sryza/aas

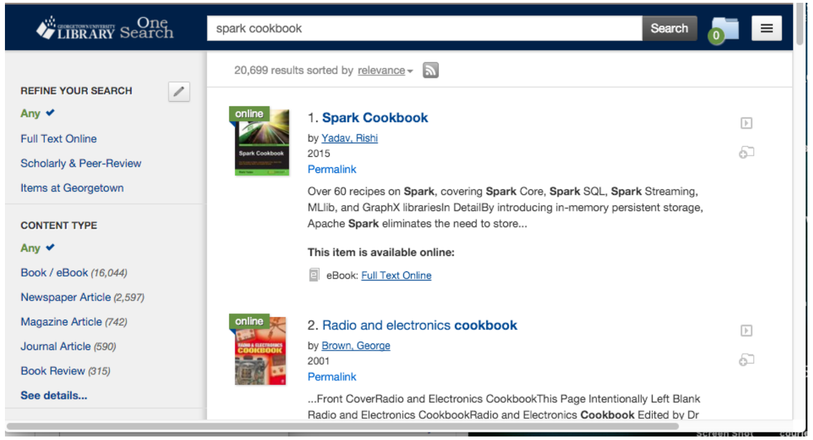
PACKT — I got these on sale for $5 each.
You can get them for free at library.georgetown.edu











Some web resources

Dean Wampler's Spark Workshop

- https://github.com/deanwampler/spark-workshop
- Scala
- 12 demos/exercised:
  - *Intro, WordCount (2), Matrix math, Web Crawler, Inverted Index, n-gram calculations, Joins, SparkSQL, Hive, and SparkStreaming.*

Databricks Reference Apps

- Log Analysis Application — monitor Apache access logs in batch & streaming
- Twitter Streaming Language Classifier — Spark MLLib, classifying , filtering & clustering
- Weather TimeSeries Data Application with Cassandra
  - *https://github.com/databricks/reference-apps*
  - *https://www.gitbook.com/book/databricks/databricks-spark-reference-applications/details*

Questions?

Spark Programming and Mini Tutorial

Two ways to run Spark:
1. Command line (pyspark or ipyspark); 2. Stand-alone scripts

Install on Linux VM with:

```
$ sudo easy_install ipython==1.2.1
```

Run pyspark with IPython:

```
$ PYSPARK_DRIVER_PYTHON=ipython pyspark
```

Check your version of PySpark:

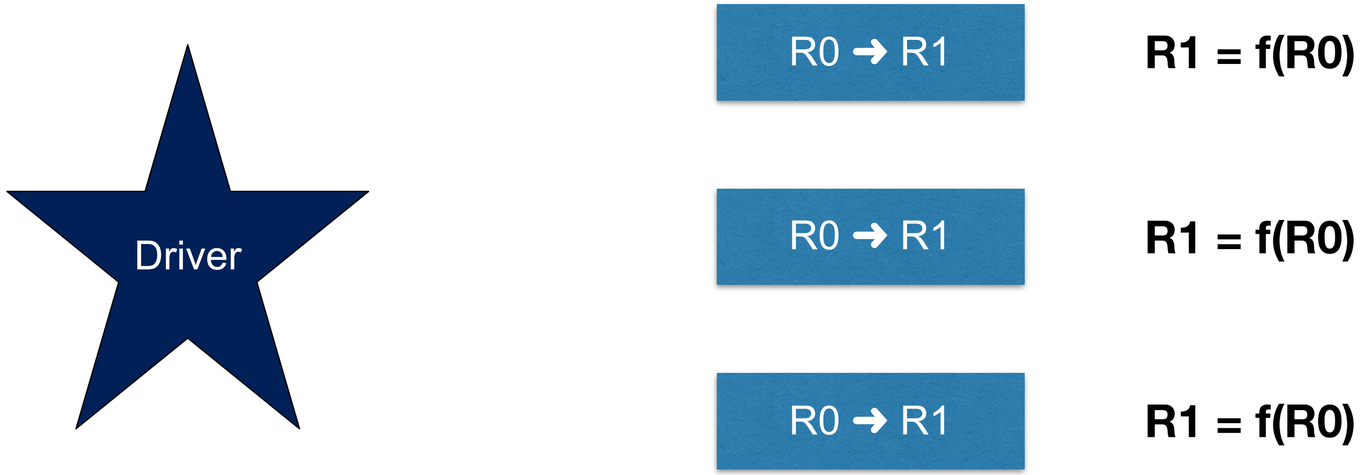
```
In [1]: sc.version
```

bootstrap.sh does this automatically now!

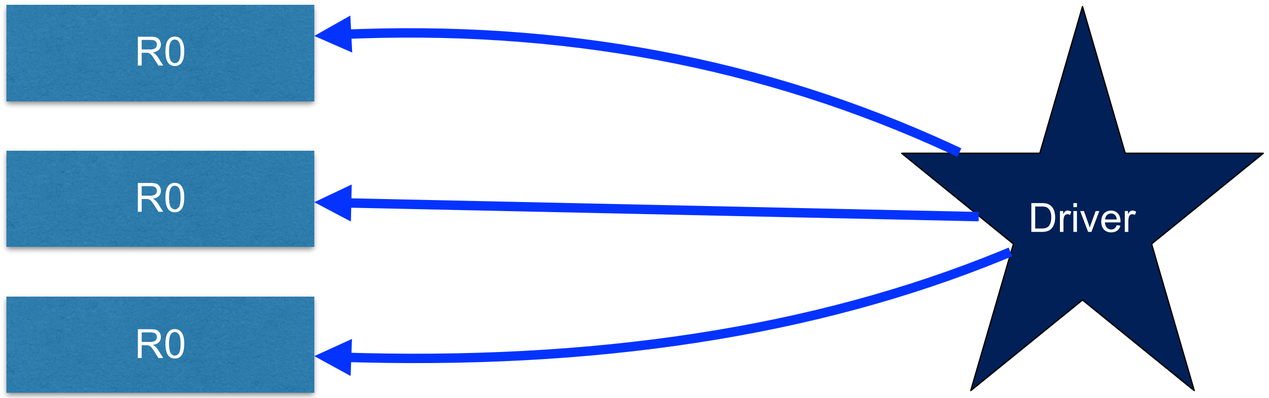
$ ipyspark to launch with ipython

RDD methods can be *actions* or *transformations*

Transformations create new RDDs within the worker nodes

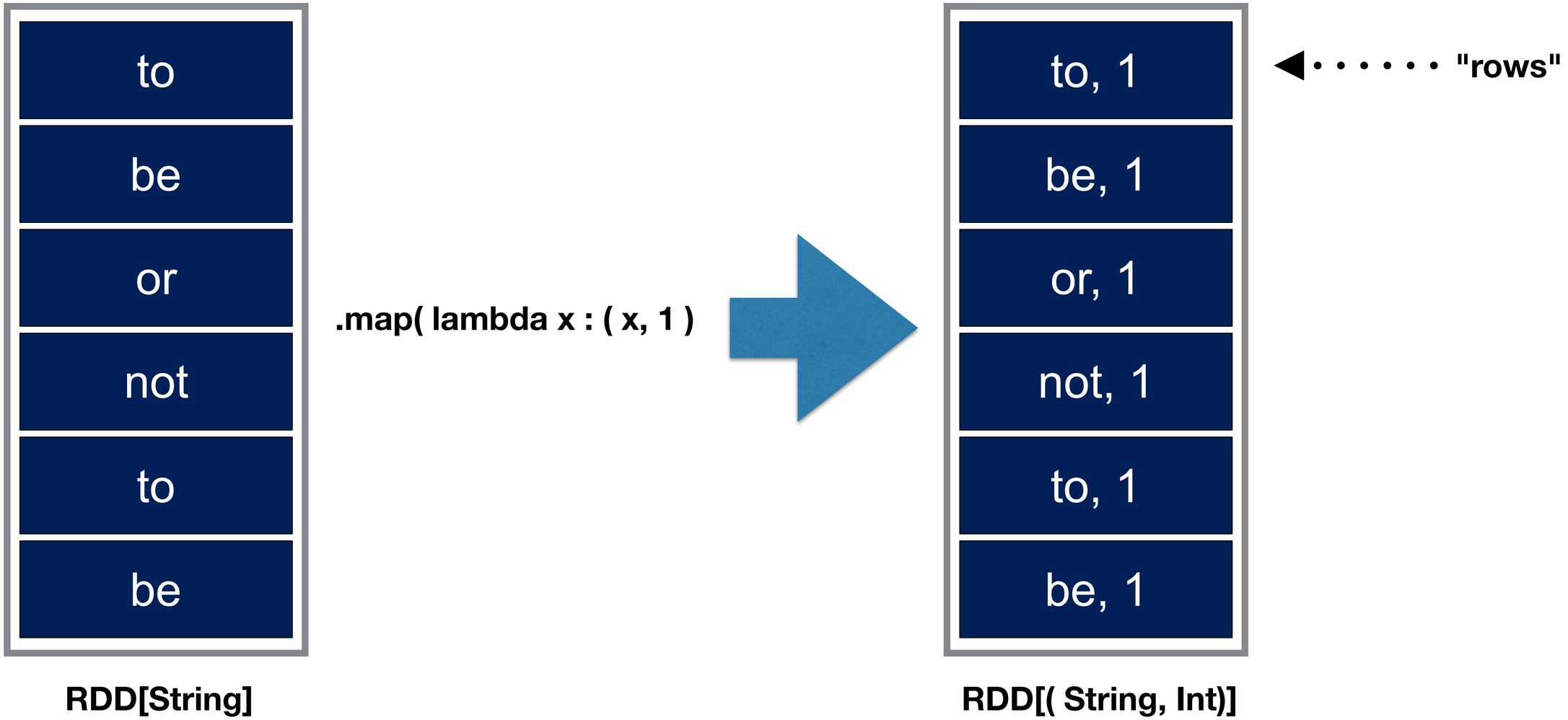


Actions move data between the worker nodes and the Driver:



Spark *transformations* produce RDDs (sometimes from other RDDs)

.map() is the same as mrjob's map function:



There are many transformations:

Operate on any RDD:

- S = R.map(f) — *f should return a value*
- S = R.flatMap(f) — *same, but f should return a Seq*
- S = R.filter(f) — *f returns boolean True/False*
- S = R.mapPartitions(f)
- S = R.mapPartitionsWithIndex(f) — *f is called with (index,val)*
- S = R.sample(withReplacement, fraction, seed) — *Produces a random sampling*
- S = R.union(R2) — *Produces union of two RDDs*
- S = R.intersection(R2) — *Intersection of two RDD2*

Operate on RDDs of (k,v) pairs:

- S = R.groupByKey([numTasks])
- S = R.reduceByKey(func, [numTasks]) — *f must take (v1,v2)*
- S = R.aggregateByKey(zeroValue)
- S = R.sortByKey([ascending])
- S = R.join(R2, [numTasks])
  - https://spark.apache.org/docs/1.3.0/programming-guide.html

Spark *actions* return values to the driver program

Operate on any RDD:

- R.reduce(f) = f(f(f(a,b), c), d), e)… => val
- R.collect( ) = RDD values
- R.count() = # of elements in RDD
- R.first() = first element in RDD; same as .take(1)
- R.take(n) = first n elements
- R.takeSample(withReplacement, n [, seed]) = random sampling
  - *Use withReplacement=False to avoid repeats.*
- RDD.saveAsTextFile(dirname) = Writes elements to HDFS

Operate on RDDs of (k,v) pairs:

- RDD.countByKey() = Returns a histogram (e.g. (k, count of k))

- https://spark.apache.org/docs/1.3.0/programming-guide.html

Spark commands for making RDDs and controlling the Spark Application

sc is the SparkContext.

- R = sc.parallelize(*alist*) — *Turns alist into an RDD*
- R = sc.range(*start, end=None, step=1, numSlices=None*) — *Creates an RDD of ints*
- R = sc.sequenceFile(*path, ...*)
- R = sc.textFile(*path*) — *Read a text file from HDFS, local file system, S3, etc...*
- R = wholeTextFiles(*path, minPartitions=None, use_unicode=True*) — *Read a directory of text files.*

- R = sc.union(R1, R2, R3, ...) — *Combine RDDs*

- sc.addFile(path) — *Add a file to every Spark node*
- pyspark.SparkFiles.get() — *Gets the files that were added*

Other features

Caching — Memory/CPU trade-off.

Broadcast variables — Send data to all nodes

Accumulators — Similar to Hadoop *counters*

Tuning Spark — http://spark.apache.org/docs/latest/tuning.html

- Data Serialization
- Memory Tuning
- Tuning Data Structures
- Tuning Garbage Collection
- Parallelism — number of partitions — numPartitions=None uses the *default number of partitions*
- Data Locality

Caching keep the RDD after it has been evaluated.

Caching is good for Iterative algorithms & debugging

```
In [21]: A = sc.textFile("s3://gu-anly502/ps03/forensicswiki.2012.txt")
In [22]: B = A.filter(lambda line:"01/Jul/2012" in line)
In [23]: B.count()
...
16/02/28 20:03:53 INFO DAGScheduler: Job 16 finished: count at <ipython-input-23-9628edc95825>:1, took 68.834135 s
Out[23]: 35039

In [24]: B.count()
...
16/02/28 20:05:45 INFO DAGScheduler: Job 17 finished: count at <ipython-input-24-9628edc95825>:1, took 75.764237 s
Out[24]: 35039

In [25]: B.count()
...
16/02/28 20:16:25 INFO DAGScheduler: Job 18 finished: count at <ipython-input-25-9628edc95825>:1, took 72.602255 s
Out[25]: 35039

In [26]: B.cache()
...
Out[26]: PythonRDD[24] at RDD at PythonRDD.scala:43

In [27]: B.count()
...
16/02/28 20:17:34 INFO DAGScheduler: Job 19 finished: count at <ipython-input-27-9628edc95825>:1, took 69.611443 s
Out[27]: 35039

In [28]: B.count()
...
16/02/28 20:17:36 INFO DAGScheduler: Job 20 finished: count at <ipython-input-28-9628edc95825>:1, took 1.639699 s
Out[28]: 35039
```

Spark applications — standalone scripts

Scripts are run with spark-submit:

```
#!/usr/bin/spark-submit
#
# wordcount as a pyspark

import sys
from operator import add
from pyspark import SparkContext
```

You must create the SparkContext:

```
from pyspark import SparkContext
sc     = SparkContext( appName="PythonWordCount" )
```

Beware — syntax errors don't always appear on EMR

```
$ ./wordcount.py s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz

$ ./wordcount.py s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz

$ python wordcount.py
  File "wordcount.py", line 26
    counts = lines.map(lambda line: filter(str.isalpha,line))) \
                                                              ^
SyntaxError: invalid syntax

$
```

Python2 vs. Python3

sc.textFile() returns *unicode* text.

- Unicode text and strings are different in Python2, but not Python3
- Under Python2, our filtering, top-10 wordcount looks like this:

```
##
## Run WordCount on Spark
##

sc     = SparkContext( appName="PythonWordCount" )
lines  = sc.textFile( filename, 1 )
counts = lines.flatMap(lambda line: line.split(' ')) \
              .map(lambda word: filter(unicode.isalpha,word)) \
              .map(lambda x: (x, 1)) \
              .reduceByKey(add)

top20counts = counts.sortBy(lambda x: x[1], ascending=False) \
              .take(20)
for (word, count) in top20counts:
    print "%-10s: %i" % (word, count)

##
## Terminate the Spark job
##

sc.stop()
```

A new module to parse Apache web logs (with Spark):

```
#!/usr/bin/env python2
#
# https://databricks.com/blog/2015/04/21/analyzing-apache-access-logs-with-databricks-cloud.html

import re              # bring in regular expression package
import dateutil, dateutil.parser
from pyspark.sql import Row
APPACHE_COMBINED_LOG_REGEX = '([(\d\.)]+) [^ ]+ [^ ]+ \[(.*)\] "(.*)" (\d+) [^ ]+ ("(.*)")? ("(.*)")?'
WIKIPAGE_PATTERN = "(index.php\?title=|/wiki/)([^ &]*)"

appache_re  = re.compile(APPACHE_COMBINED_LOG_REGEX)
wikipage_re = re.compile(WIKIPAGE_PATTERN)

def parse_apache_log_line(logline):
    m = appache_re.match(logline)
    if m==None:
        raise Error("Invalid logline: {}".format(logline))

    timestamp = m.group(2)
    request   = m.group(3)
    agent     = m.group(7).replace('"','') if m.group(7) else ''

    request_fields = request.split(" ")
    url         = request_fields[1] if len(request_fields)>2 else ""
    datetime    = dateutil.parser.parse(timestamp.replace(":", " ", 1)).isoformat()
    (date,time) = (datetime[0:10],datetime[11:])

    n = wikipage_re.search(url)
    wikipage = n.group(2) if n else ""

    return Row(
        ipaddr = m.group(1),
        timestamp = timestamp,
        request = request,
        result = int(m.group(4)),
        user = m.group(5),
        referrer = m.group(6),
        agent = agent,
        url = url,
        datetime = datetime,
        date = date,
        time = time,
        wikipage = wikipage)
```

A *Row* object is return

Using the Apache module:

Need to get module to every python instance:

```
$ ipyspark --py-files sweblog.py
...
In [2]: import sweblog
In [3]: lines = sc.textFile("s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz")
In [4]: plines = lines.map(sweblog.parse_apache_log_line).cache()
In [5]: plines.count()
Out[5]: 52677

In [6]: plines.take(1)
Out[6]: [Row(agent='', date='2012-01-13', datetime='2012-01-13T00:05:10-08:00', ipaddr=u'80.243.191.178', referrer=u'http://
www.forensicswiki.org/" "Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 6.0)', request=u'GET / HTTP/1.0', result=301,
time='00:05:10-08:00', timestamp=u'13/Jan/2012:00:05:10 -0800', url=u'/', user=u'"http://www.forensicswiki.org/" "Mozilla/4.0
(compatible; MSIE 7.0b; Windows NT 6.0)"', wikipage='')]

In [7]: plines.map(lambda x: ( x.result, 1 )).reduceByKey(operator.add).collect()

Out[7]:
[(200, 47381),
 (301, 330),
 (302, 383),
 (304, 4275),
 (403, 2),
 (404, 284),
 (206, 17),
 (503, 4),
 (500, 1)]
```

Using the Apache module:

Need to get module to every python instance:

```
$ ipyspark --py-files sweblog.py
...
In [2]: import sweblog
In [3]: lines = sc.textFile("s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz")
In [4]: plines = lines.map(sweblog.parse_apache_log_line).cache()
In [5]: plines.count()
Out[5]: 52677

In [6]: plines.take(1)
Out[6]: [Row(agent='', date='2012-01-13', datetime='2012-01-13T00:05:10-08:00', ipaddr=u'80.243.191.178', referrer=u'http://
www.forensicswiki.org/" "Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 6.0)', request=u'GET / HTTP/1.0', result=301,
time='00:05:10-08:00', timestamp=u'13/Jan/2012:00:05:10 -0800', url=u'/', user=u'"http://www.forensicswiki.org/" "Mozilla/4.0
(compatible; MSIE 7.0b; Windows NT 6.0)"', wikipage='')]

In [7]: plines.map(lambda x: ( x.result, 1 )).reduceByKey(operator.add).collect()

Out[7]:
[(200, 47381),
 (301, 330),
 (302, 383),
 (304, 4275),
 (403, 2),
 (404, 284),
 (206, 17),
 (503, 4),
 (500, 1)]
```

Sort on the driver node:

```
sorted(plines.map(lambda x: ( x.result, 1 ))\
       .reduceByKey(operator.add).collect())
```

Using the Apache module:

Need to get module to every python instance:

```
$ ipyspark --py-files sweblog.py
...
In [2]: import sweblog
In [3]: lines = sc.textFile("s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz")
In [4]: plines = lines.map(sweblog.parse_apache_log_line).cache()
In [5]: plines.count()
Out[5]: 52677

In [6]: plines.take(1)
Out[6]: [Row(agent='', date='2012-01-13', datetime='2012-01-13T00:05:10-08:00', ipaddr=u'80.243.191.178', referrer=u'http://
www.forensicswiki.org/" "Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 6.0)', request=u'GET / HTTP/1.0', result=301,
time='00:05:10-08:00', timestamp=u'13/Jan/2012:00:05:10 -0800', url=u'/', user=u'"http://www.forensicswiki.org/" "Mozilla/4.0
(compatible; MSIE 7.0b; Windows NT 6.0)"', wikipage='')]

In [7]: plines.map(lambda x: ( x.result, 1 )).reduceByKey(operator.add).collect()

Out[7]:
[(200, 47381),
 (301, 330),
 (302, 383),
 (304, 4275),
 (403, 2),
 (404, 284),
 (206, 17),
 (503, 4),
 (500, 1)]
```

Sort on the driver node:

```
sorted(plines.map(lambda x: ( x.result, 1 ))\
       .reduceByKey(operator.add).collect())
```

Sort with .takeOrdered():

```
plines.map(lambda x: ( x.result, 1 ))\
      .reduceByKey(operator.add).takeOrdered(100)
```

Using the Apache module:

Need to get module to every python instance:

```
$ ipyspark --py-files sweblog.py
...
In [2]: import sweblog
In [3]: lines = sc.textFile("s3://gu-anly502/ps03/forensicswiki/access.log.2012-01-13.gz")
In [4]: plines = lines.map(sweblog.parse_apache_log_line).cache()
In [5]: plines.count()
Out[5]: 52677

In [6]: plines.take(1)
Out[6]: [Row(agent='', date='2012-01-13', datetime='2012-01-13T00:05:10-08:00', ipaddr=u'80.243.191.178', referrer=u'http://
www.forensicswiki.org/" "Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 6.0)', request=u'GET / HTTP/1.0', result=301,
time='00:05:10-08:00', timestamp=u'13/Jan/2012:00:05:10 -0800', url=u'/', user=u'"http://www.forensicswiki.org/" "Mozilla/4.0
(compatible; MSIE 7.0b; Windows NT 6.0)"', wikipage='')]

In [7]: plines.map(lambda x: ( x.result, 1 )).reduceByKey(operator.add).collect()

Out[7]:
[(200, 47381),
 (301, 330),
 (302, 383),
 (304, 4275),
 (403, 2),
 (404, 284),
 (206, 17),
 (503, 4),
 (500, 1)]
```

Sort on the driver node:

```
sorted(plines.map(lambda x: ( x.result, 1 ))\
       .reduceByKey(operator.add).collect())
```

Sort with .takeOrdered():

```
plines.map(lambda x: ( x.result, 1 ))\
      .reduceByKey(operator.add).takeOrdered(100)
```

Sort with .sortBy():

```
plines.map(lambda x: ( x.result, 1 ))\
      .reduceByKey(operator.add)\
      .sortBy(lambda x:x[0]).collect()
```

Lab: use ipyspark

Do it now!

Data sources:

```
s3://gu-anly502/ps02/hamlet.txt
s3://gu-anly502/ps03/forensicswiki.2012.txt
s3://gu-anly502/ps03/freebase-wex-2009-01-12-articles.tsv
```

Calculate:

- # of lines in the data source
- # of forensicswiki log files in March 2012 — e.g. source.filter(lambda line:"Mar/2012" in line)

Try the Spark QuickStart and Spark Examples

- http://spark.apache.org/docs/latest/quick-start.html
- http://spark.apache.org/examples.html
- Use s3://gu-anly502/L06/README.md instead of README.md

ipython keyboard shortcuts:
^p - Previous command
^n - Next command
^f - Forward character
^b - Backward character
^a - Beginning of line
^e - End of line

https://www.pexels.com/photo/people-apple-iphone-writing-154/

Homework and L07 Preview

PS04 — Pig and Spark

Part 1 — Word Count with Pig (3 problems)

Part 2 — Analyzing ForensicsWiki logs with Pig (2 problems)

Part 3 — Spark — WordCount and Wikipedia

Extra Credit: Maxmind join with broadcast variables and the full IP address space.

Reading!

Read this Apache Spark Documentation:

- http://spark.apache.org/docs/latest/quick-start.html
- http://spark.apache.org/docs/latest/programming-guide.html
- http://spark.apache.org/docs/latest/submitting-applications.html
- http://spark.apache.org/docs/latest/cluster-overview.html
- http://minimaxir.com/2015/11/nyc-ggplot2-howto/

Skim the API

- http://spark.apache.org/docs/latest/api/python/pyspark.html
- http://spark.apache.org/docs/latest/monitoring.html

Recommended blog posts:

- http://blog.appliedinformaticsinc.com/how-to-write-spark-applications-in-python/
- http://www.mccarroll.net/blog/pyspark2/