SI 618: Large-scale distributed computation 2

Fall 2021

Instructor: Ceren Budak

Some slides courtesy of Kevyn Collins-Thompson and Yuhang Wang

Data Gathering and Processing Projects

- Proposals due next Wednesday, right before class.

- Proposal Guidelines (100 points):
  - (20 points) Summarize and motivate your proposed project.
  - (20 points) Choose and describe (at least) two different datasets.
  - (20 points) Describe how you might manipulate and join the two datasets.
  - (30 points) Describe at least three tasks you will perform to gain insights from the datasets (you can use mrjob, spark or sparksql) --- *you haven't learned sparksql yet so you can propose mrjob or spark for now.*
  - (10 points) Describe at least one visualization you might create that highlights insights you hope to gain

- You can propose an alternative project structure with prior approval (you need to contact me for this)

This week

- Hadoop Framework
- Cavium
- Spark

Hadoop Framework

- A software framework for distributed computing
- Take 100 'commodity' machines that don't share memory or disk storage
- Turn commodity machines into a cluster
    - ✓ *Redundant and Reliable*
    - ✓ *Powerful and Scalable*
    - ✓ *Cost-effective*
- Java-based APIs to Hadoop services
    - But calling these directly is tedious and error-prone so people use programming languages like spark to perform Hadoop jobs

Hadoop Framework

Major components

1. MapReduce (algorithm)
   - A programming model for large-scale data processing
2. Hadoop Distributed File System (data storage)
   - Stores and aggregates data on cluster machines
3. Hardware Architecture
   - Networked machines

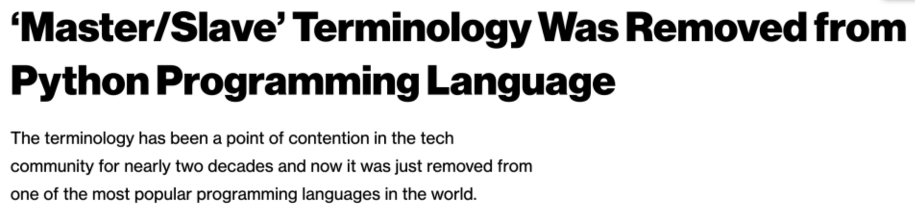
Cluster of machines running Hadoop at Yahoo! (Source: Yahoo!) via
http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

High Level Overview

Warning: The high level summary I will show right now includes unacceptable language. Sadly, this is the current state in computer science

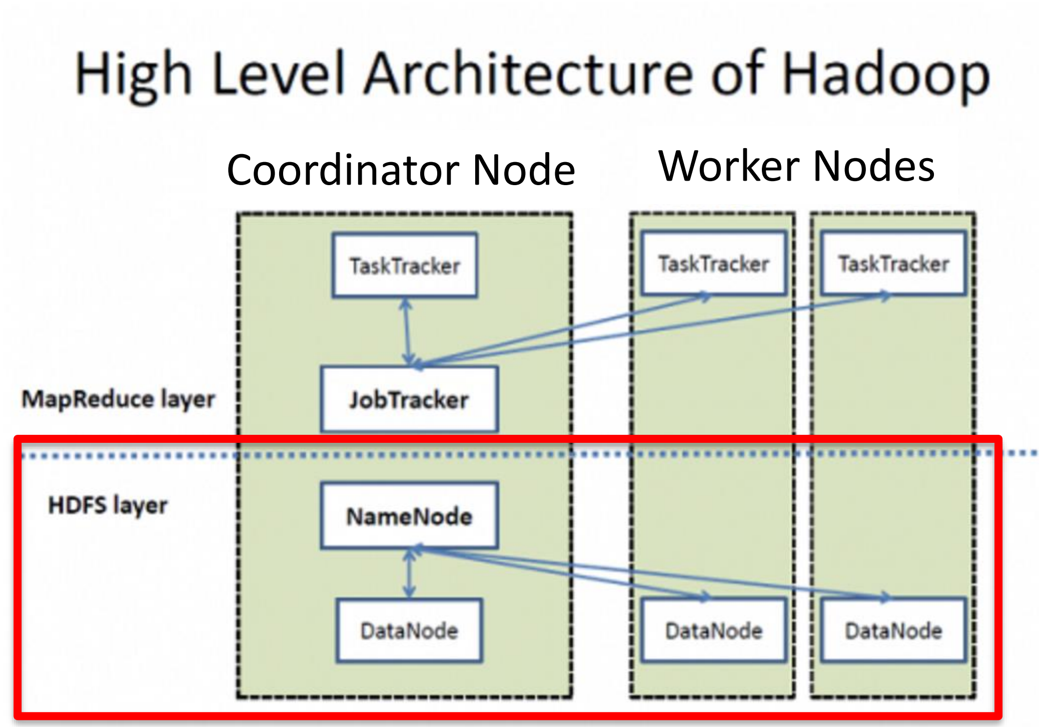
It is important for us to acknowledge the current state and highlight the fact that such terminology being considered acceptable is perhaps a function of the diversity of the teams that found it acceptable

There are efforts to change things, though.



'Master/Slave' Terminology Was Removed from Python Programming Language

The terminology has been a point of contention in the tech community for nearly two decades and now it was just removed from one of the most popular programming languages in the world.



I will focus on the HDFS aspect. For more on YARN (Job scheduling): https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Hadoop Distributed File System (HDFS)

Also need a mechanism to support the process at the data level.

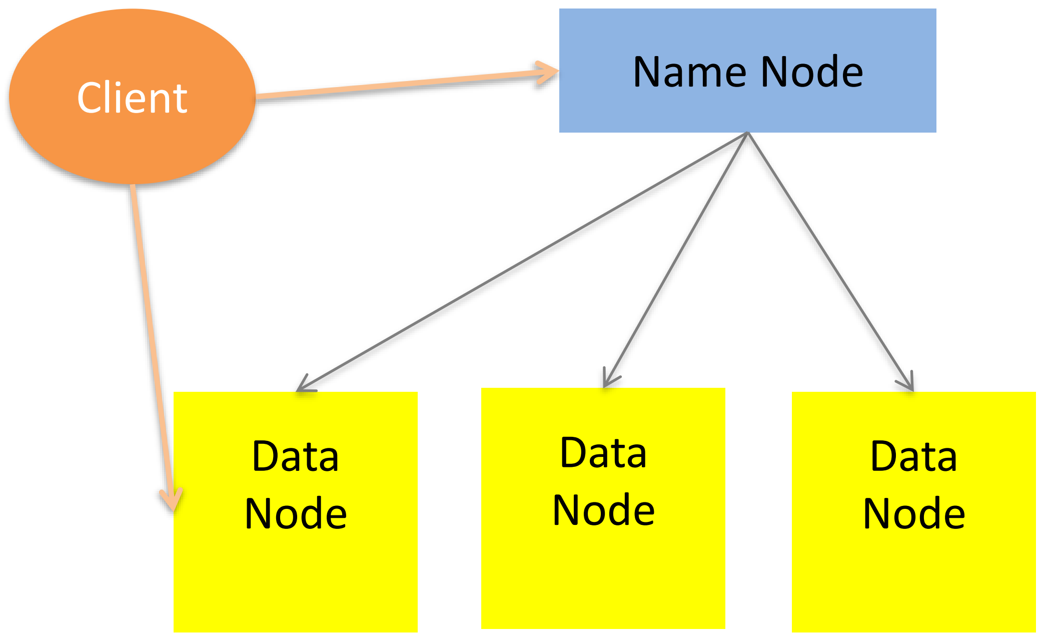
HDFS is designed to be…

- Scalable in storage and I/O bandwidth
- Highly fault-tolerant (check periodically)
- Optimized for commodity machines

Typical Settings:

- Save a file into blocks (128MB)
- Replicate 3 times

Hadoop Distributed File System (HDFS)

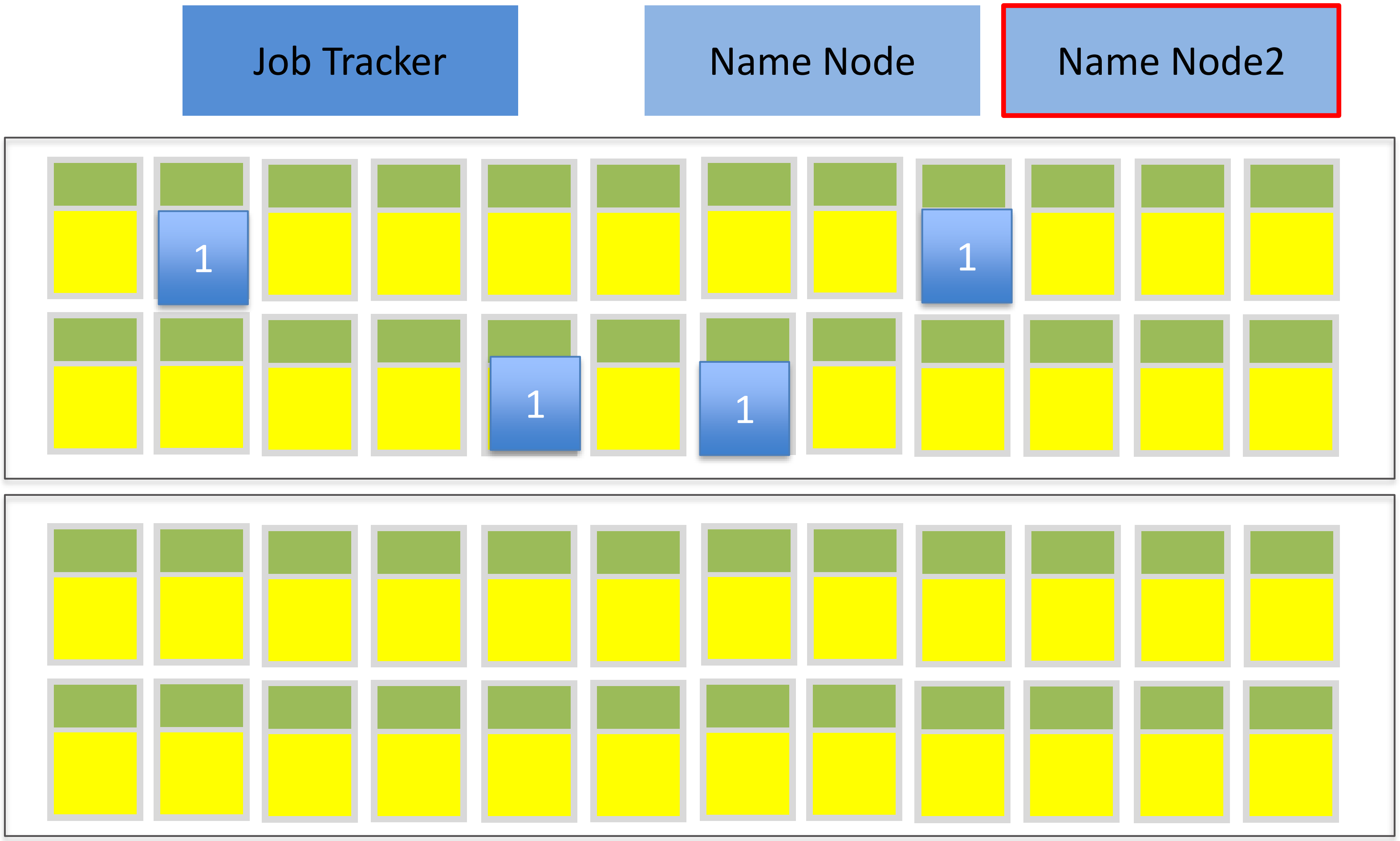


Responsible for actual read and write

Name node keeps track of where these chunks are. Name node also has a simple webpage that lists some basic details.
Single point of availability failure (which is why systems generally replicate and have two name nodes)

Hadoop MapReduce

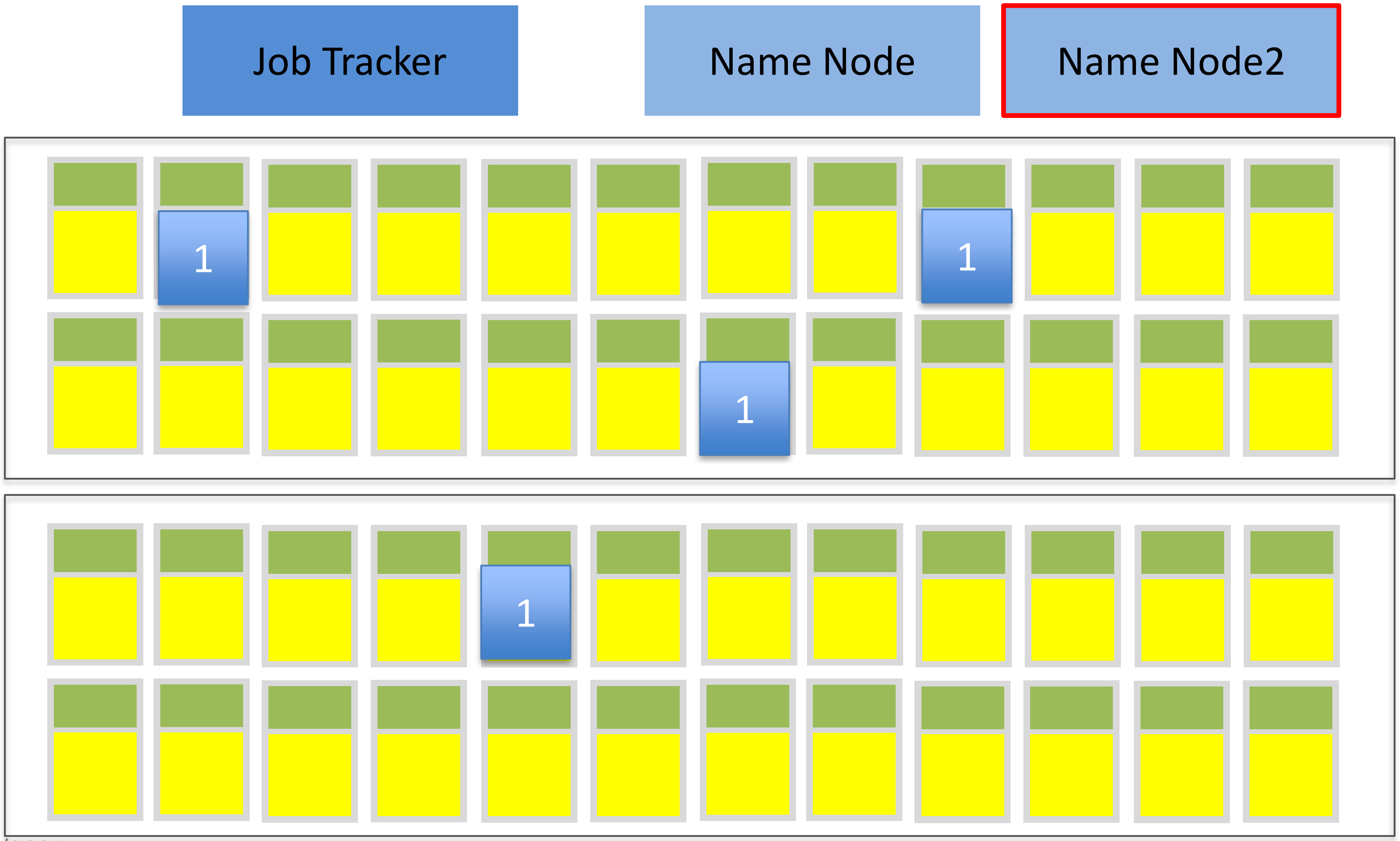
In large configurations, there might be multiple racks with multiple nodes.
Name node is rack-aware (what if an entire rack goes down? We need to be resilient to this case)



What happens if rack 1 goes offline?

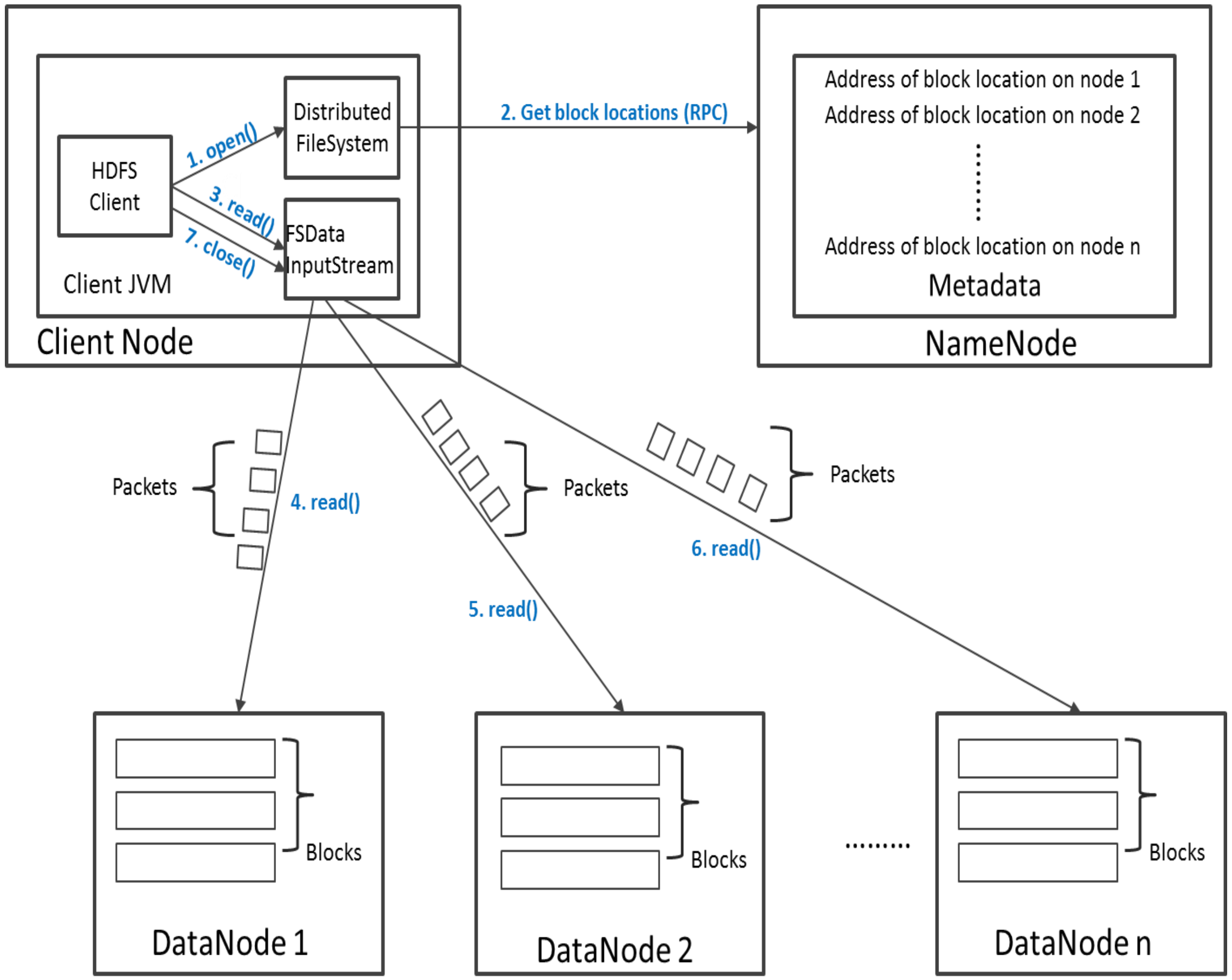
Hadoop MapReduce

In large configurations, there might be multiple racks with multiple nodes.
Name node is rack-aware (what if an entire rack goes down? We need to be resilient to this case)



What happens if rack 1 goes offline?

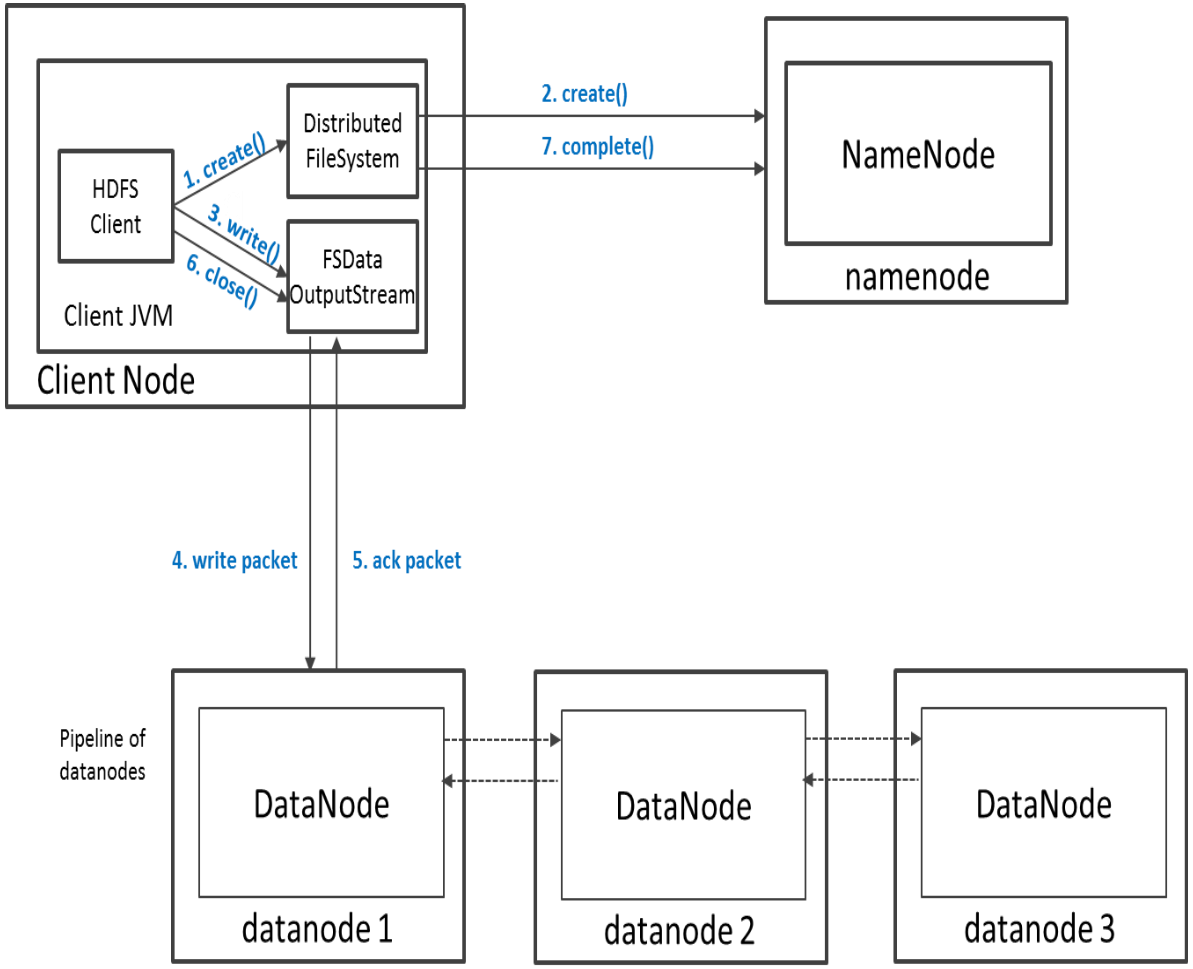
Read Operation in HDFS



Say that the client is looking for a block that is replicated on three datanodes. Namenode gives a sorted listed of datanodes to contact. Client will contact the second datanode only if it does not receive a response from the first

Why does the client go to the name node only to get the list and communicate with the data nodes directly? We don't want name node to be a single point of failure

Write Operation in HDFS



Say your replication is 3, two of those are generally saved on two different datanodes on the same rack? Why?

So far

- Hadoop: open-source framework for reliable, scalable, distributed computing.
  1. MapReduce (algorithm)
     - A programming model for large-scale data processing
  2. Hadoop Distributed File System (data storage)
     - Stores and aggregates data on cluster machines
  3. Hardware Architecture
     - Networked machines
- Yet, we did everything locally. Now we go to the cluster!

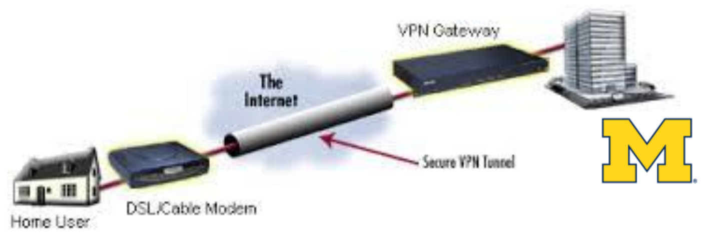
Cavium

- The Cavium Thunder X Hadoop cluster is a next-generation Hadoop cluster available to researchers at the University of Michigan.

- It holds 3PB of storage for researchers to approach and analyze data science problems.
- The cluster consists of 40 servers each containing 96 ARMv8 cores and 512GB of RAM per server.
- The cluster provides 1PB of total disk space, 40GbE inter-node networking, and Hadoop 2.7.2 with Spark 2 and Hive 2.

- ARC-TS provides various other computing solutions. Check out: https://arc-ts.umich.edu/systems-services/ for more info.
- We will use the Cavium Hadoop cluster in this class. For the user guide, please check: https://arc-ts.umich.edu/cavium/user-guide/

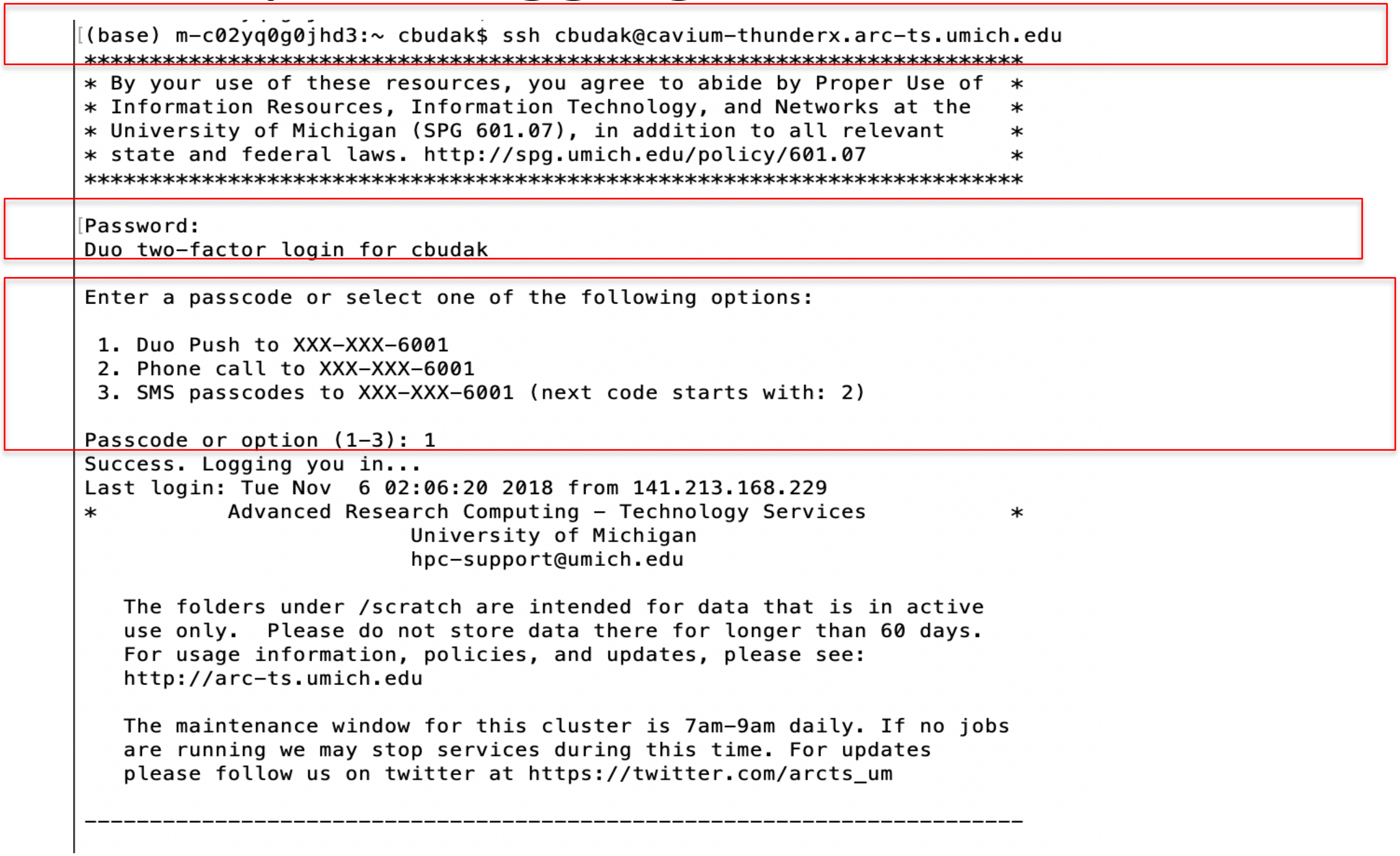
Working off-campus:
Step 1: Use VPN (Virtual Private Network)



- VPN: uses encryption and tunneling to establish secure private connections over third-party networks

- See http://www.itcom.itd.umich.edu/vpn/ for details on how to install VPN software for UM.

Step 2: Logging in to remote server



```
(base) m-c02yq0g0jhd3:~ cbudak$ ssh cbudak@cavium-thunderx.arc-ts.umich.edu
**************************************************************************
* By your use of these resources, you agree to abide by Proper Use of    *
* Information Resources, Information Technology, and Networks at the     *
* University of Michigan (SPG 601.07), in addition to all relevant       *
* state and federal laws. http://spg.umich.edu/policy/601.07             *
**************************************************************************

Password:
Duo two-factor login for cbudak

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-6001
 2. Phone call to XXX-XXX-6001
 3. SMS passcodes to XXX-XXX-6001 (next code starts with: 2)

Passcode or option (1-3): 1
Success. Logging you in...
Last login: Tue Nov  6 02:06:20 2018 from 141.213.168.229
*          Advanced Research Computing - Technology Services             *
                         University of Michigan
                         hpc-support@umich.edu

   The folders under /scratch are intended for data that is in active
   use only.  Please do not store data there for longer than 60 days.
   For usage information, policies, and updates, please see:
   http://arc-ts.umich.edu

   The maintenance window for this cluster is 7am-9am daily. If no jobs
   are running we may stop services during this time. For updates
   please follow us on twitter at https://twitter.com/arcts_um

------------------------------------------------------------------------
```

You can also use software such as Putty (on Windows machines).

Working on a remote server: copying files between your laptop and the remote account

- Option 1: Use scp
  - Copy from local to server:
    - scp localfile uniqname@cavium-thunderx.arc-ts.umich.edu:remot
      (copy a file)
    - scp -r localdir uniqname@cavium-thunderx.arc-ts.umich.edu:remotedir   (copy an entire directory)
  - Copy from server to local
    - scp uniqname@cavium-thunderx.arc-ts.umich.edu:remotefile loca
- Option 2: Use CyberDuck (on Mac) or WinSCP (on Windows)

Unix command line refresher

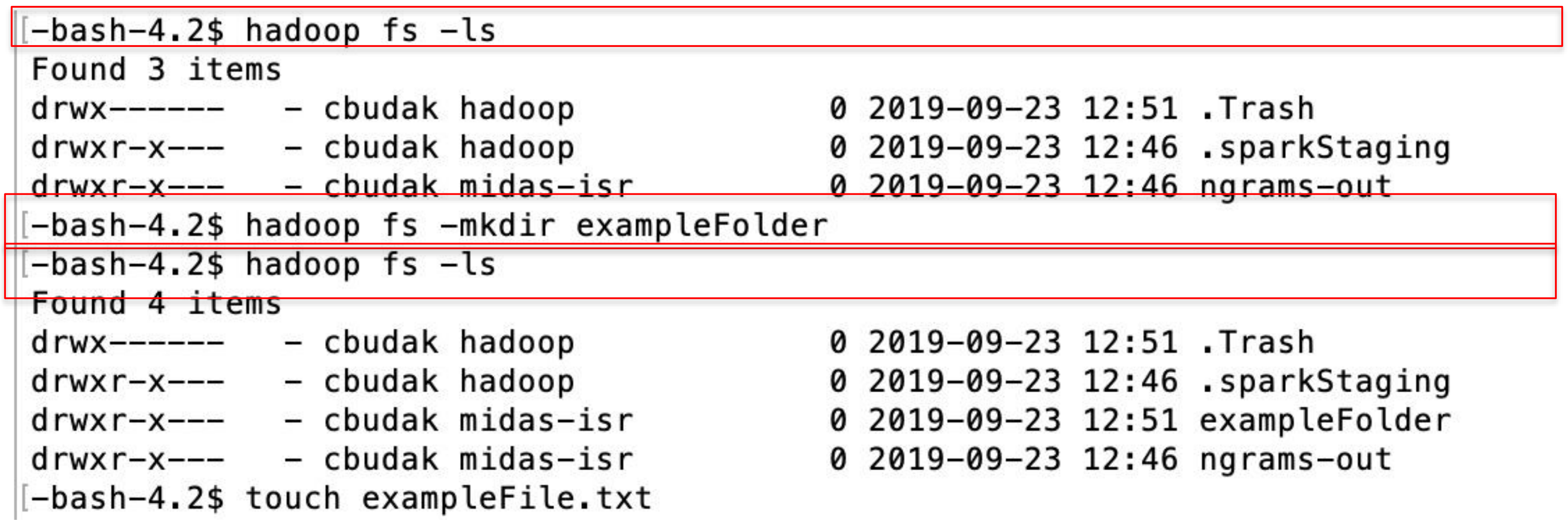
- `ls  : list contents of a directory`
- `pwd : print working directory`
- `Paths`
  `/home/<youruniqname>`
- `Special directory names:`
  ```
  .    Current directory
  ..   Parent directory
  ```
- `cd <directory>  : change directory`
- `cat <filename> : dump contents of a file`
- `rm <filename> : remove a file`

Hadoop file system follows the Unix command line conventions

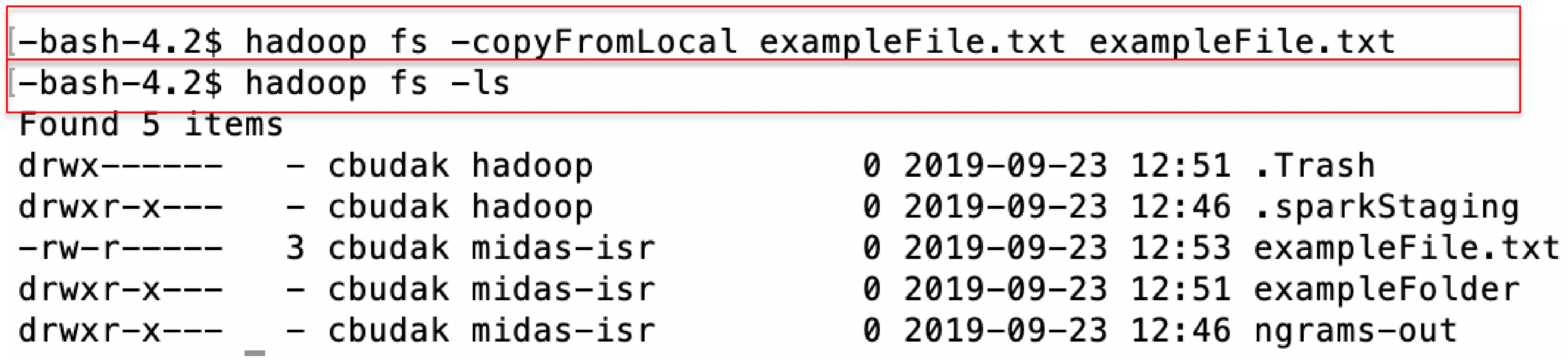
- `hadoop fs -`**`ls`**
- `hadoop fs -`**`ls <directory>`**
- `hadoop fs -`**`cat <filename>`**
- `hadoop fs -`**`rm <filename>`**

HDFS Practice

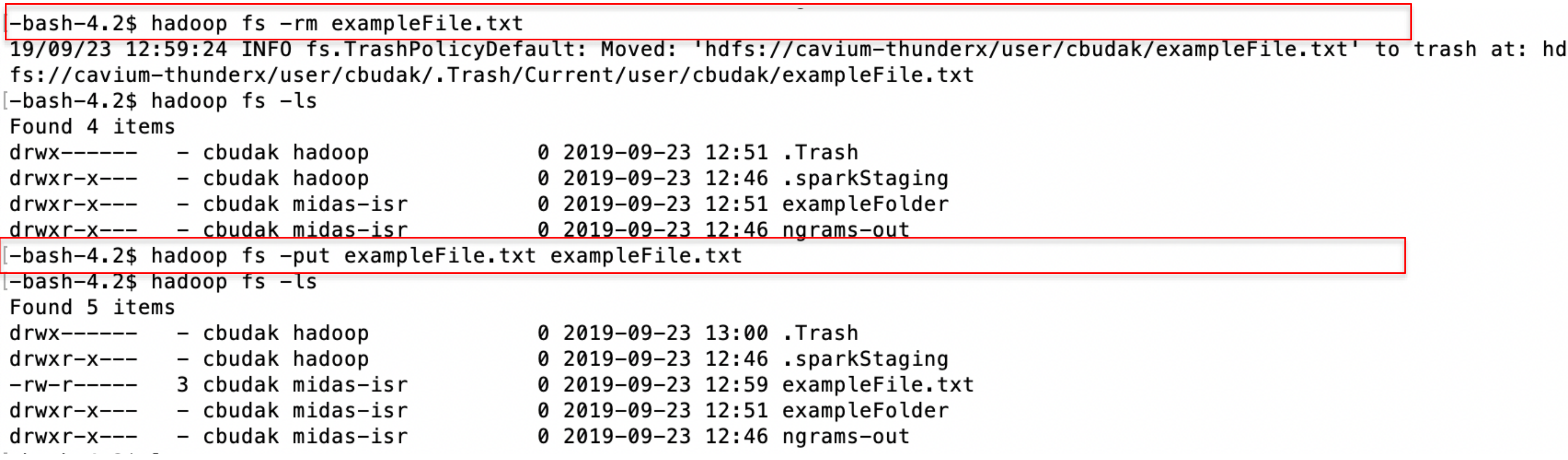
- Let’s try some simple commands together…



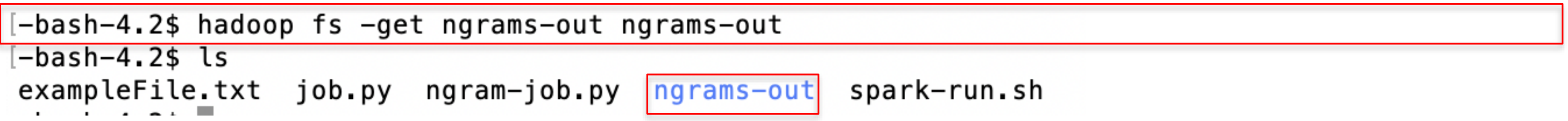
```
-bash-4.2$ hadoop fs -ls
Found 3 items
drwx------   - cbudak hadoop             0 2019-09-23 12:51 .Trash
drwxr-x---   - cbudak hadoop             0 2019-09-23 12:46 .sparkStaging
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:46 ngrams-out
-bash-4.2$ hadoop fs -mkdir exampleFolder
-bash-4.2$ hadoop fs -ls
Found 4 items
drwx------   - cbudak hadoop             0 2019-09-23 12:51 .Trash
drwxr-x---   - cbudak hadoop             0 2019-09-23 12:46 .sparkStaging
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:51 exampleFolder
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:46 ngrams-out
-bash-4.2$ touch exampleFile.txt
```



```
-bash-4.2$ hadoop fs -copyFromLocal exampleFile.txt exampleFile.txt
-bash-4.2$ hadoop fs -ls
Found 5 items
drwx------   - cbudak hadoop             0 2019-09-23 12:51 .Trash
drwxr-x---   - cbudak hadoop             0 2019-09-23 12:46 .sparkStaging
-rw-r-----   3 cbudak midas-isr          0 2019-09-23 12:53 exampleFile.txt
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:51 exampleFolder
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:46 ngrams-out
```

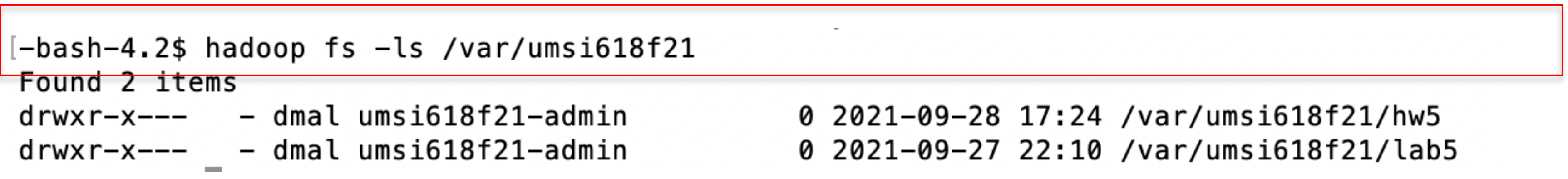


```
-bash-4.2$ hadoop fs -rm exampleFile.txt
19/09/23 12:59:24 INFO fs.TrashPolicyDefault: Moved: 'hdfs://cavium-thunderx/user/cbudak/exampleFile.txt' to trash at: hd
fs://cavium-thunderx/user/cbudak/.Trash/Current/user/cbudak/exampleFile.txt
-bash-4.2$ hadoop fs -ls
Found 4 items
drwx------   - cbudak hadoop             0 2019-09-23 12:51 .Trash
drwxr-x---   - cbudak hadoop             0 2019-09-23 12:46 .sparkStaging
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:51 exampleFolder
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:46 ngrams-out
-bash-4.2$ hadoop fs -put exampleFile.txt exampleFile.txt
-bash-4.2$ hadoop fs -ls
Found 5 items
drwx------   - cbudak hadoop             0 2019-09-23 13:00 .Trash
drwxr-x---   - cbudak hadoop             0 2019-09-23 12:46 .sparkStaging
-rw-r-----   3 cbudak midas-isr          0 2019-09-23 12:59 exampleFile.txt
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:51 exampleFolder
drwxr-x---   - cbudak midas-isr          0 2019-09-23 12:46 ngrams-out
```



```
-bash-4.2$ hadoop fs -get ngrams-out ngrams-out
-bash-4.2$ ls
exampleFile.txt  job.py  ngram-job.py  ngrams-out  spark-run.sh
```

This is the folder where we will share data with you



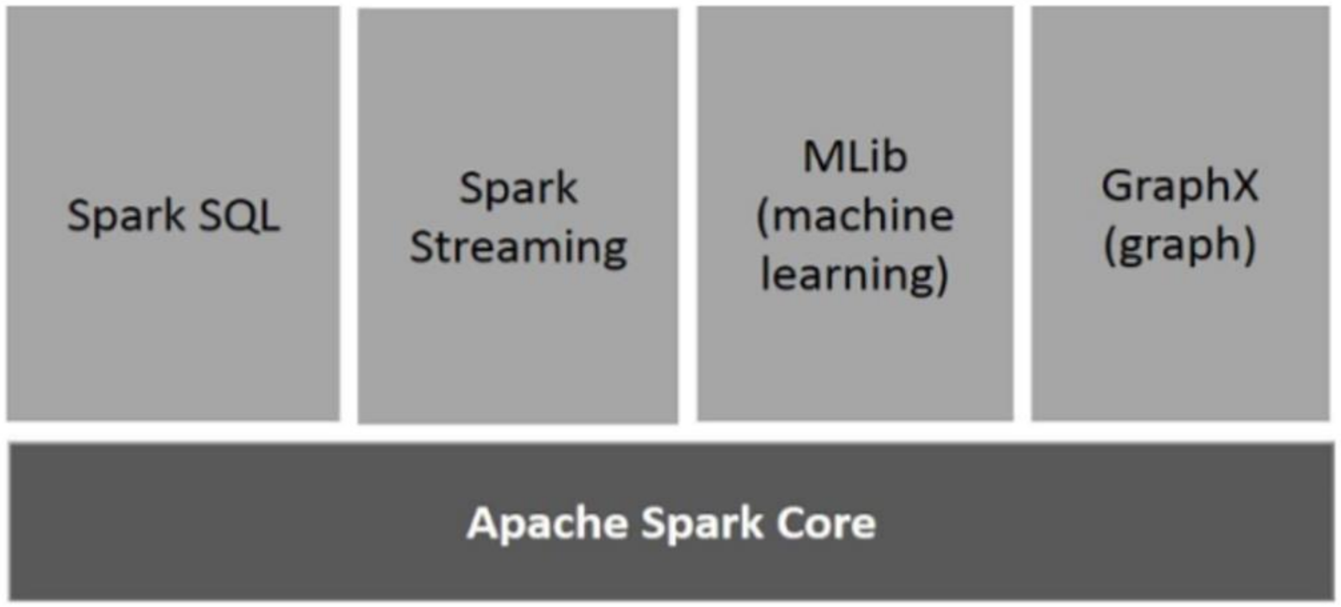
```
-bash-4.2$ hadoop fs -ls /var/umsi618f21
Found 2 items
drwxr-x---   - dmal umsi618f21-admin          0 2021-09-28 17:24 /var/umsi618f21/hw5
drwxr-x---   - dmal umsi618f21-admin          0 2021-09-27 22:10 /var/umsi618f21/lab5
```

Spark - Fast and general engine for large-scale data processing

- Up to 100 times faster than Hadoop MapReduce
- Written in Scala, providing Scala, Java and Python APIs
- Supports both batch mode and real-time data stream processing
- Write once, run everywhere
  - Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, S3.

http://spark.apache.org/docs/latest/quick-start.html
Sources: https://spark.apache.org, http://en.wikipedia.org/wiki/Apache_Spark

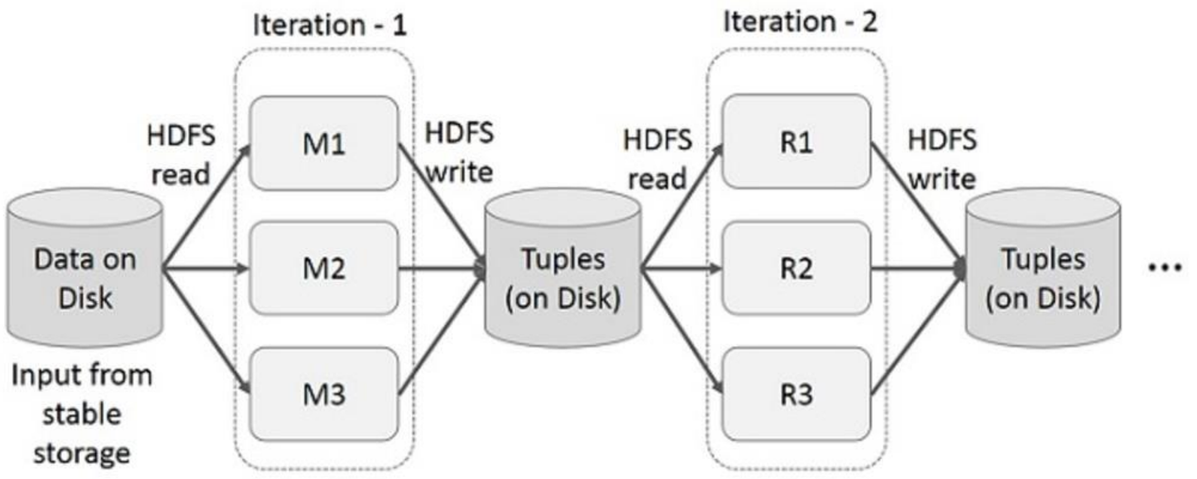
Spark Components



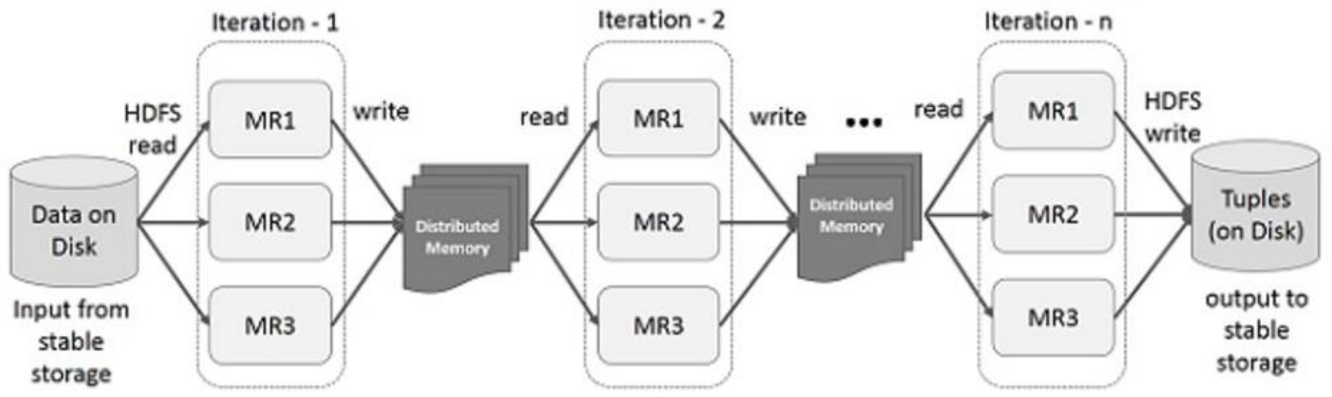
https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

Traditional MapReduce vs. Spark: iterative operations

MapReduce



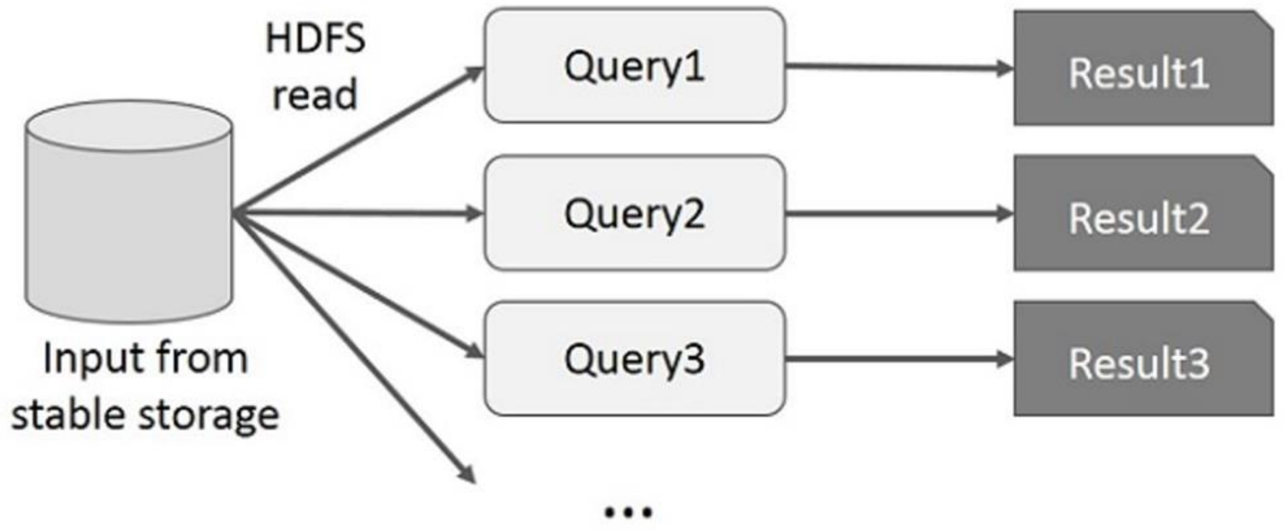
Spark



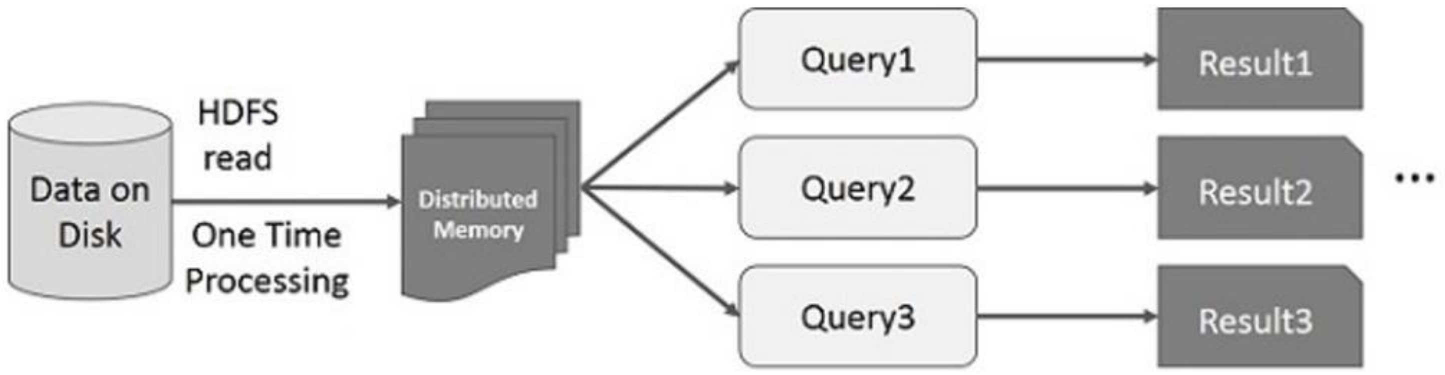
https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

Traditional MapReduce vs. Spark: interactive operations

MapReduce



Spark

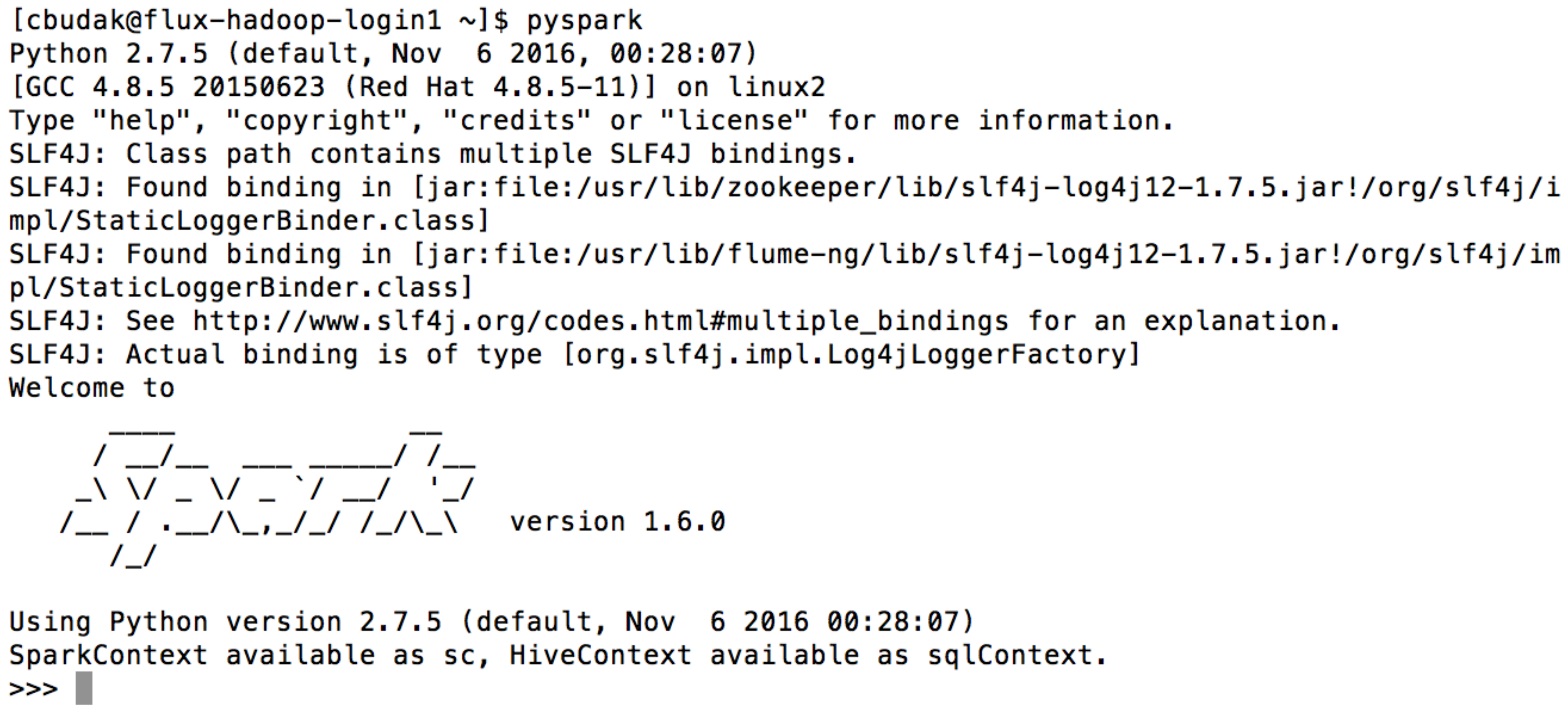


https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

Basic Spark Concepts

- SparkContext
  - represents the connection to a Spark cluster, and can be used to create RDDs, accumulators and broadcast variables on that cluster.
- Resilient Distributed Dataset (RDD)
  - RDDs have <u>actions</u>, which return values,
  - and <u>transformations</u>, which return new RDDs
- Two code execution modes:
  - Interactive Python shell: PySpark
  - Running standalone applications: spark-submit

Pyspark: an interactive Python shell running on top of Spark

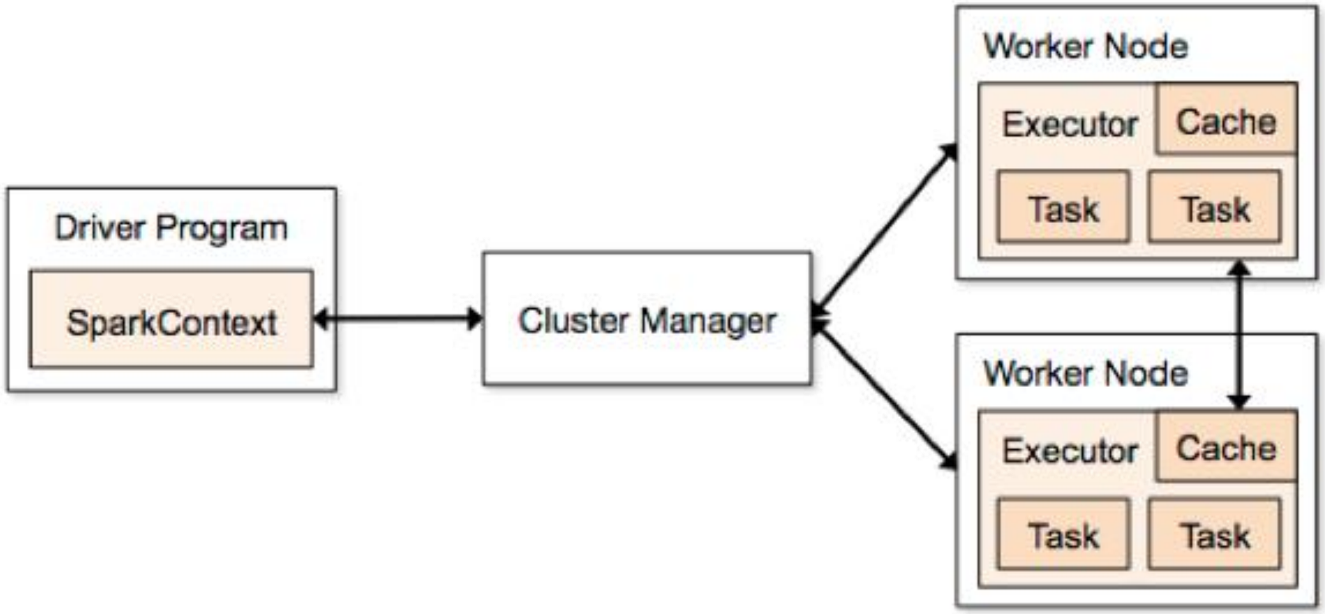


```
[cbudak@flux-hadoop-login1 ~]$ pyspark
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/im
pl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.7.5 (default, Nov  6 2016 00:28:07)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

SparkContext

- Represents a connection to a computing cluster
- In PySpark, a SparkContext is auto-created for you in a variable called `sc`



Source: https://spark.apache.org/docs/1.1.1/cluster-overview.html

Resilient Distributed Dataset (RDD)

- The fundamental abstraction for distributed data computation
- An RDD is:
  - A immutable (read-only) collection of items
  - Partitioned across computing nodes
  - That can be manipulated in parallel
- Once you have a SparkContext you can create RDDs
- Example: A collection of lines from a text file

```
>>> lines = sc.textFile("input.txt") # Create an RDD called lines, input needs to be on HDFS
>>> lines.count() # Count the number of items in this RDD
3
>>> lines.first() # First item in this RDD, i.e. first line of input.txt
u'summer school 2012 in indiana'
```

In Spark, all work involves one of three kinds of operations on RDDs

1. Creating new RDDs

e.g. `lines = sc.textFile("input.txt")`

Input: various. Output: new RDD

2. Transforming existing RDDs

e.g. `pigLines = lines.filter(lambda line: "pig" in line)`

Input: one or more RDDs. Output: new RDD

3. Computing actions on RDDs to get a result

e.g. `pigLines.first()`

Input: one or more RDDs. Output: various non-RDD

Three kinds of operations on RDDs

1. **Creating new RDDs**

e.g. **`lines = sc.textFile("input.txt")`**

Input: various. Output: new RDD

2. Transforming existing RDDs

e.g. `pigLines = lines.filter(lambda line: "pig" in line)`

Input: one or more RDDs. Output: new RDD

3. Computing actions on RDDs to get a result

e.g. `pigLines.first()`

Input: one or more RDDs. Output: various non-RDD

Creation methods

1. Load data from external storage
   – `lines = sc.textFile('./nyt/1985.txt’)`

2. Parallelize an existing data structure
   – `lines = sc.parallelize(["pandas", "i like pandas"])`

Three kinds of operations on RDDs

1. <u>Creating</u> new RDDs

e.g. `lines = sc.textFile("input.txt")`

Input: various. Output: new RDD

2. <u>Transforming</u> existing RDDs

e.g. `pigLines = lines.filter(lambda line: "pig" in line)`

Input: one or more RDDs. Output: new RDD

3. Computing <u>actions</u> on RDDs to get a result

e.g. `pigLines.first()`

Input: one or more RDDs. Output: various non-RDD

Element-wise transformations: **map** and filter

- The map() transformation takes in a function and applies it to each element in the RDD with the result of the function being the new value of each element in the resulting RDD.

```
nums = sc.parallelize([1, 2, 3, 4])
squared = nums.map(lambda x: x * x).collect()
for num in squared:
    print "%i " % (num)
```

- The magic of Spark: operations like ‘map’ are parallelized across the cluster.

Digression: Lambda Function

```
nums = sc.parallelize([1, 2, 3, 4])
squared = nums.map(lambda x: x * x).collect()
for num in squared:
    print "%i " % (num)
```

You can create anonymous functions in Python using lambda. For instance instead of:

```
def multiply(x, y):
    return x * y
```

You can have

```
r = lambda x, y: x * y
r(12, 3)   # call the lambda function
```

You also do not have to assign lambda function to a variable:

```
(lambda x, y: x * y)(3,4)
```

Element-wise transformations: map and **filter**

- The filter() transformation takes in a function and returns an RDD that only has elements that pass the filter() function.

```
pigLines = lines.filter(lambda line: "pig" in line)

Or

def containsError(s):
    return "error" in s
word = rdd.filter(containsError)
```

Element-wise transformations: flatMap

Sometimes we want to produce **multiple output elements for each input element**. The operation to do this is called flatMap(). As with map(), the function we provide to flatMap() is called individually for each element in our input RDD. Instead of returning a single element, we return an iterator with our return values. Rather than producing an RDD of iterators, we get back an RDD that consists of the elements from all of the iterators.

```
lines = sc.parallelize(["hello world", "hi"])
words = lines.flatMap(lambda line: line.split(" "))
words.first()  # returns "hello"
```

Set-like transformations

```
errorsRDD = inputRDD.filter(lambda x: "error" in x)
warningsRDD = inputRDD.filter(lambda x: "warning" in x)
badLinesRDD = errorsRDD.union(warningsRDD)

rdd1.union(rdd2)
rdd1.intersection(rdd2)
rdd1.subtract(rdd2)
rdd1.distinct()
```

Pair RDDs: collections of (key, value) pairs

- **`reduceByKey(func)`**
- **`sortByKey()`**
- `groupByKey()`
- `mapValues(func)`
- `flatMapValues(func)`
- `keys()`
- `values()`

Pair RDDs: sortByKey()

Just like the Python sorted() function: sort (key, value) pairs alphabetically

```
word_counts_sorted = word_counts.sortByKey()
```

Pair RDDs: sortBy()

Just like the Python sorted() using a key= function

```
word_counts_sorted = word_count3.sortBy(lambda x: x[1],
ascending = False)
```

Pair RDDs: reduceByKey(func)

Takes a function that operates on the values of two elements with the same key and returns a new RDD.

```
sumRDD = rdd.reduceByKey(lambda
x, y: x + y)
```

Pair RDDs: reduce(func)

Reduce is an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program

```
sumRDD = rdd.reduce(lambda x, y:
x + y)
```

Three kinds of operations on RDDs

1. Creating new RDDs

e.g. `lines = sc.textFile("input.txt")`

Input: various. Output: new RDD

2. Transforming existing RDDs

e.g. `pigLines = lines.filter(lambda line: "pig" in line)`

Input: one or more RDDs. Output: new RDD

3. Computing actions on RDDs to get a result

e.g. `pigLines.first()`

Input: one or more RDDs. Output: various non-RDD

Actions

- `take(`*n*`)`   Gets the first k elements

```
for line in lines.take(10):
  print line
```

- `collect()`   Gets **all** elements
- Take an RDD and turn it into a local list
- Be careful when using on a large RDD

- `first()` Gets the first element

These operations create non-RDD objects. Be careful not to apply before you need to (it will take a while)

RDD operations we have learned:

- map(*f*, *preservesPartitioning=False*)
- flatMap(*f*, *preservesPartitioning=False*)
- filter(*f*)
- sortBy(*keyfunc*, *ascending=True*, *numPartitions=None*)
- reduceByKey(*func*, *numPartitions=None*)
- take(*num*)
- collect()
- saveAsTextFile(*path*, *compressionCodecClass=None*)

More about Spark

- Basic statistic of RDDs:
  - max(*key=None*)
  - min(*key=None*)
  - mean()
  - count()
  - sum()
  - stdev()
- How to join RDDs:
  - join(*other*, *numPartitions=None*)
  - leftOuterJoin(*other*, *numPartitions=None*)
  - rightOuterJoin(*other*, *numPartitions=None*)
  - fullOuterJoin(*other*, *numPartitions=None*)

https://spark.apache.org/docs/1.3.0/api/python/pyspark.html#pyspark.RDD

Resources

- Spark Documentation
  - https://spark.apache.org/docs/latest/quick-start.html
  - https://spark.apache.org/docs/latest/programming-guide.html
  - https://spark.apache.org/docs/latest/api/python/index.html
- Spark on Cavium
  - https://arc-ts.umich.edu/cavium/user-guide/

Example: Word Count

Let’s work on it together!