CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2019
Doc 16 Spark, Cluster, AWS EMR
Mar 26, 2019

Showing Spark Operations

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
        .appName("Counter") \
        .getOrCreate()
spark.sparkContext
df = spark.range(15)
df.show()
```

```
+---+
| id|
+---+
|  0|
|  1|
|  2|
|  3|
|  4|
|  5|
|  6|
|  7|
|  8|
|  9|
| 10|
| 11|
| 12|
| 13|
| 14|
+---+
```

Run in Notebook

```
counter = 0
def count(item):
    global counter
    print("item: ", item.id, "counter: ", counter)
    counter = counter + 1

from pyspark.sql import SparkSession
spark = SparkSession.builder \
        .appName("Counter") \
        .getOrCreate()
spark.sparkContext

print("start")
df = spark.range(16)
smaller = df.coalesce(4)
smaller.foreach(count)
print("end")
print(counter)
```

Output in Notebook

```
start
end
0
```

Output at Command line

```
item:  12 counter:  0
item:  13 counter:  1
item:  14 counter:  2
item:  15 counter:  3
item:  4 counter:  0
item:  5 counter:  1
item:  6 counter:  2
item:  7 counter:  3
item:  8 counter:  0
item:  9 counter:  1
item:  10 counter:  2
item:  11 counter:  3
item:  0 counter:  0
item:  1 counter:  1
item:  2 counter:  2
item:  3 counter:  3
```

Towards AWS

Need a program

Issues

- Packaging files
- Running in local cluster of one machine
- Logging
- File references

Sample Program

printExample.py

```
from __future__ import print_function
def print5000():
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .master("local") \
        .appName("Print") \
        .getOrCreate()
    print(spark.range(5000).selectExpr("sum(id)").collect())

if __name__ == "__main__":
        print5000()
```

Running in Temp Spark Runtime

I put the SPARK_HOME/bin & SPARK_HOME/sbin on my path

Set SPARK_HOME

setenv SPARK_HOME /Java/spark-2.4.0-bin-hadoop2.7

->spark-submit ./printExample.py

2019-03-24 21:08:58 WARN NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

2019-03-24 21:08:59 INFO SparkContext:54 - Running Spark version 2.4.0

2019-03-24 21:08:59 INFO SparkContext:54 - Submitted application: Print

2019-03-24 21:08:59 INFO SecurityManager:54 - Changing view acls to: whitney

2019-03-24 21:08:59 INFO SecurityManager:54 - Changing modify acls to: whitney

2019-03-24 21:08:59 INFO SecurityManager:54 - Changing view acls groups to:

2019-03-24 21:08:59 INFO SecurityManager:54 - Changing modify acls groups to:

Output

99 lines

```
2019-03-24 21:09:03 INFO  TaskSetManager:54 - Finished task 0.0 in stage 0.0 (TID 0) i
localhost (executor driver) (1/1)
2019-03-24 21:09:03 INFO  TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks
completed, from pool
2019-03-24 21:09:03 INFO  DAGScheduler:54 - ResultStage 0 (collect at /Users/whitney/
Spring19/sparkExamples/./printExample.py:9) finished in 0.201 s
2019-03-24 21:09:03 INFO  DAGScheduler:54 - Job 0 finished: collect at /Users/whitney/
Spring19/sparkExamples/./printExample.py:9, took 0.247842 s
[Row(sum(id)=12497500)]
2019-03-24 21:09:03 INFO  SparkContext:54 - Invoking stop() from shutdown hook
2019-03-24 21:09:03 INFO  AbstractConnector:318 - Stopped Spark@17df1968{HTTP/1
{0.0.0.0:4041}
2019-03-24 21:09:03 INFO  SparkUI:54 - Stopped Spark web UI at http://192.168.0.112:4
2019-03-24 21:09:03 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerM
stopped!
2019-03-24 21:09:03 INFO  MemoryStore:54 - MemoryStore cleared
2019-03-24 21:09:03 INFO  BlockManager:54 - BlockManager stopped
2019-03-24 21:09:03 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
```

Line 86 → **[Row(sum(id)=12497500)]**

File input/output

Hardcoding I/O file names in source not desirable

parseExample.py

```
    def files_from_args():
        import argparse
        parser = argparse.ArgumentParser()
        parser.add_argument('-i', '--input', default='input')
        parser.add_argument('-o', '--output',default='output')
        args = parser.parse_args()
        return (args.input, args.output)

if __name__ == "__main__":
        inputfile, outputfile = files_from_args()
        print("input = " + inputfile)
        print("output = " + outputfile)
```

Example Usage

->python parseExample.py -i cat
input = cat
output = output

->python parseExample.py -i cat -output dog
usage: parseExample.py [-h] [-i INPUT] [-o OUTPUT]
printExample.py: error: unrecognized arguments: dog

->python parseExample.py -i cat --output dog
input = cat
output = dog

->python parseExample.py --output dog -i cat
input = cat
output = dog

Sample Program

```
def write5000(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Write") \
        .getOrCreate()

    spark.range(5000).selectExpr('id *2').write.format('csv').save(file)
    spark.stop()

def files_from_args():
        import argparse
        parser = argparse.ArgumentParser()
        parser.add_argument('-i', '--input', default='input')
        parser.add_argument('-o', '--output',default='output')
        args = parser.parse_args()
        return (args.input, args.output)

if __name__ == "__main__":
        _, outputfile = files_from_args()
        write5000(outputfile)
```

->spark-submit ./writeExample.py

2019-03-24 21:45:50 WARN NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

2019-03-24 21:45:51 INFO SparkContext:54 - Running Spark version 2.4.0

2019-03-24 21:45:51 INFO SparkContext:54 - Submitted application: Write

2019-03-24 21:45:51 INFO SecurityManager:54 - Changing view acls to: whitney

2019-03-24 21:45:51 INFO SecurityManager:54 - Changing modify acls to: whitney

2019-03-24 21:45:51 INFO SecurityManager:54 - Changing view acls groups to:

2019-03-24 21:45:51 INFO SecurityManager:54 - Changing modify acls groups to:

Starting a Spark Cluster of One

Command SPARK_HOME/sbin/start-master.sh

>start-master.sh

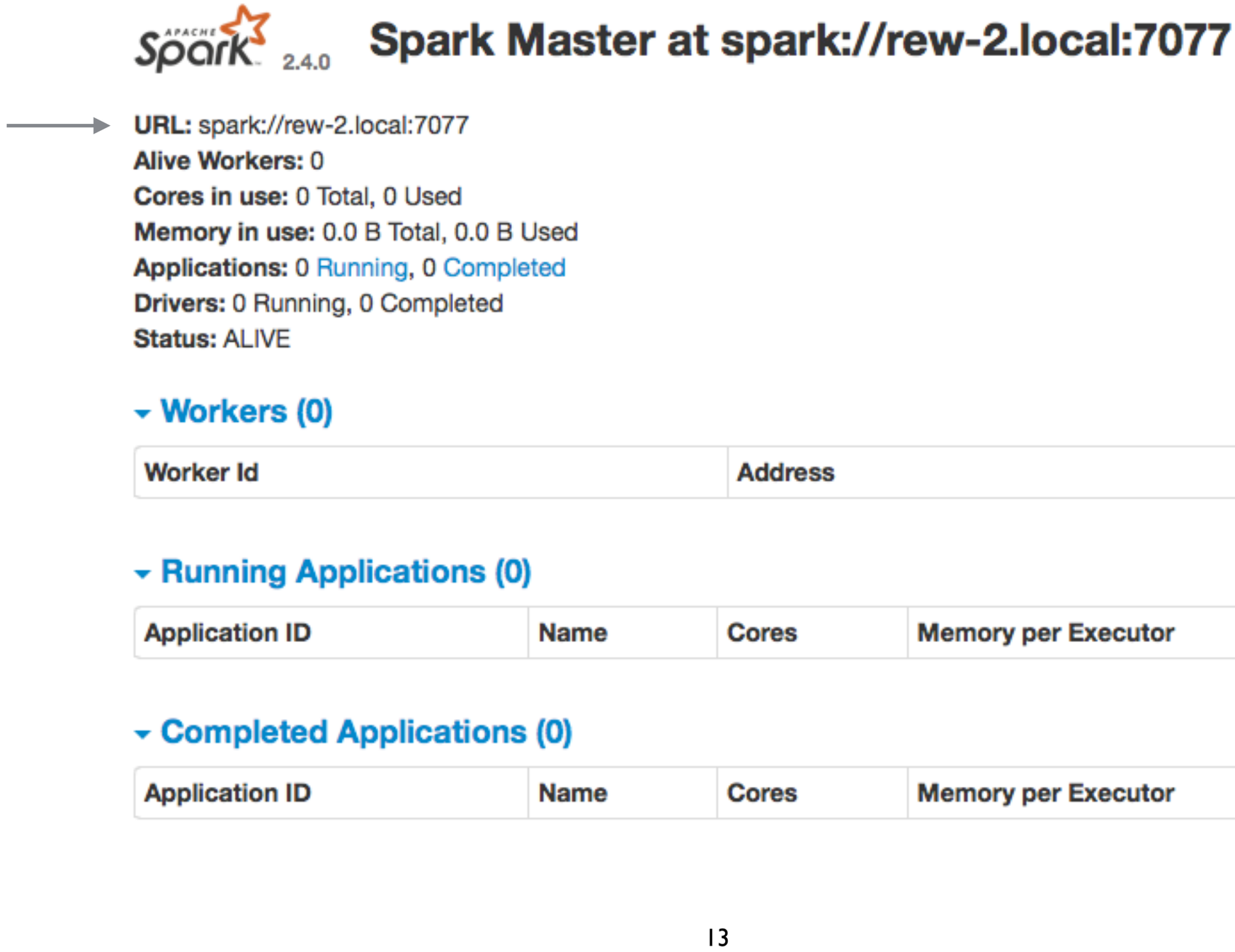
starting org.apache.spark.deploy.master.Master, logging to /Java/spark-2.4.0-bin-hadoop2.7/logs/spark-whitney-org.apache.spark.deploy.master.Master-1-rew-2.local.out

Master Web Page

localhost:8080

127.0.0.1:8080

0.0.0.0:8080



Starting slave on local machine

Command SPARK_HOME/sbin/start-slave.sh

->start-slave.sh spark://rew-2.local:7077

Master Web Page

Spark Master at spark://rew-2.local:7077

URL: spark://rew-2.local:7077
Alive Workers: 1
Cores in use: 8 Total, 0 Used
Memory in use: 15.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State
worker-20190324215103-192.168.0.112-64770	192.168.0.112:64770	ALIVE

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time

Submitting Job to Spark on Cluster

run SPARK_HOME/bin/spark-submit

->spark-submit --master spark://rew-2.local:7077 ./printExample.py -o sampleOut

Master Web Page

Spark Master at spark://rew-2.local:7077

URL: spark://rew-2.local:7077
Alive Workers: 1
Cores in use: 8 Total, 0 Used
Memory in use: 15.0 GB Total, 0.0 B Used
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190324215103-192.168.0.112-64770	192.168.0.112:64770	ALIVE	8 (0 Used)	15.0 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190324215330-0000	Write	6	1024.0 MB	2019/03/24 21:53:30	whitney	FINISHED	5 s

Application Page

Application: Write

ID: app-20190324215330-0000
Name: Write
User: whitney
Cores: Unlimited (8 granted)
Executor Limit: Unlimited (1 granted)
Executor Memory: 1024.0 MB
Submit Date: 2019/03/24 21:53:30
State: FINISHED

Executor Summary (1)

ExecutorID	Worker	Cores	Memory	State	Logs

Removed Executors (1)

ExecutorID	Worker	Cores	Memory	State	Logs
0	worker-20190324215103-192.168.0.112-64770	8	1024	KILLED	stdout stderr

Starting/Stopping Master/Slave

Commands in SPARK_HOME/sbin

->start-master.sh

->start-slave.sh spark://air-6.local:7077

->stop-master.sh

->stop-slave.sh

->start-all.sh

->stop-all.sh

spark-submit

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

Spark Properties

name
master
logging
memory
etc

https://spark.apache.org/docs/latest/configuration.html

name - displayed in Spark Master Web page

master

Master URL	Meaning
local	Run Spark locally with one worker thread.
local[K]	Run Spark locally with K worker threads
local[K,F]	Run Spark locally with K worker threads and F maxFailures
local[*]	Run Spark locally with as many worker threads as logical cores on your machine.
local[*,F]	Run Spark locally with as many worker threads as logical cores on your machine and F maxFailures.
spark://HOST:PORT	Connect to the given Spark standalone cluster master.
spark:// HOST1:PORT1,HOST2:PORT2	Connect to the given Spark standalone cluster with standby masters with Zookeeper.
mesos://HOST:PORT	Connect to the given Mesos cluster.
yarn	Connect to a YARN cluster in client or cluster mode

Examples

Start spark master-slave using default value

```
->spark-submit  pythonCode.py
```

Start spark master-slave using all cores

```
->spark-submit --master "local[*]" pythonCode.py
```

Submit job to existing master

```
->spark-submit --master spark://air-6.local:7077 \
                pythonCode.py
```

Adding additional Python code

Add Python.zip, .egg or .py files using the flag

--py-files

Setting Properties

In precedence order

In program

submit command

config file

Setting master in Code

```
def print5000():
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .master("local") \
        .appName("Print") \
        .getOrCreate()
print(spark.range(5000).selectExpr("sum(id)").collect())
```

Don't set master in code

It overrides value in command line and config file

So will not be able change master settings without recompiling

Warning

```
def write5000(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Write") \
        .getOrCreate()

    spark.range(5000).selectExpr('id *2').write.format('csv').save(file)
    spark.stop()
```

Spark will not override existing files

If you run this a second time without removing files you get an exception

But You Can Override the Default

```
def write5000(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Write") \
        .getOrCreate()

    spark.range(5000).selectExpr('id *2').write.mode("overwrite").format('csv').save(file)
    spark.stop()
```

Issue - Debugging

Debugger not available for program running on cluster

Print statements

- Don't count on seeing them from slaves

Logging

- Spark uses log4j 1.2

Sample

```
def write5000(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Write") \
        .getOrCreate()
    log4jLogger = spark.sparkContext._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger(__name__)
    LOGGER.info("Start")
    spark.range(5000).selectExpr('id *2').write.mode("overwrite").format('csv').save(file)
    LOGGER.error("End")
    spark.stop()
```

Some of the Output

2019-03-25 19:35:40 INFO StateStoreCoordinatorRef:54 - Registered StateStoreCoordinator e

2019-03-25 19:35:40 INFO __main__:? - Start

2019-03-25 19:35:41 INFO CoarseGrainedSchedulerBackend$DriverEndpoint:54 - Registered executor NettyRpcEndpointRef(spark-client://Executor) (192.168.0.112:53755) with ID 0
2019-03-25 19:35:41 INFO BlockManagerMasterEndpoint:54 - Registering block manager 192.168.0.112:53757 with 366.3 MB RAM, BlockManagerId(0, 192.168.0.112, 53757, None)
2019-03-25 19:35:43 INFO FileOutputCommitter:108 - File Output Committer Algorithm version is 1
2019-03-25 19:35:43 INFO SQLHadoopMapReduceCommitProtocol:54 - Using output committer class org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2019-03-25 19:35:43 INFO CodeGenerator:54 - Code generated in 196.113223 ms
2019-03-25 19:35:43 INFO SparkContext:54 - Starting job: save at NativeMethodAccessorImpl.java:0
2019-03-25 19:35:43 INFO DAGScheduler:54 - Got job 0 (save at NativeMethodAccessorImpl.java:0) with 2 output partitions
2019-03-25 19:35:43 INFO DAGScheduler:54 - Final stage: ResultStage 0 (save at NativeMethodAccessorImpl.java:0)
2019-03-25 19:35:43 INFO DAGScheduler:54 - Parents of final stage: List()
2019-03-25 19:35:43 INFO DAGScheduler:54 - Missing parents: List()
2019-03-25 19:35:43 INFO DAGScheduler:54 - Submitting ResultStage 0 (MapPartitionsRDD[2] at save at NativeMethodAccessorImpl.java:0), which has no missing parents
2019-03-25 19:35:43 INFO MemoryStore:54 - Block broadcast_0 stored as values in memory (estimated size 149.5 KB, free 366.2 MB)
2019-03-25 19:35:43 INFO MemoryStore:54 - Block broadcast_0_piece0 stored as bytes in memory (estimated size 54.5 KB, free 366.1 MB)
2019-03-25 19:35:43 INFO BlockManagerInfo:54 - Added broadcast_0_piece0 in memory on 192.168.0.112:53752 (size: 54.5 KB, free: 366.2 MB)
2019-03-25 19:35:43 INFO SparkContext:54 - Created broadcast 0 from broadcast at DAGScheduler.scala:1161
2019-03-25 19:35:43 INFO DAGScheduler:54 - Submitting 2 missing tasks from ResultStage 0 (MapPartitionsRDD[2] at save at NativeMethodAccessorImpl.java:0) (first 15 tasks are for partitions Vector(0, 1))
2019-03-25 19:35:43 INFO TaskSchedulerImpl:54 - Adding task set 0.0 with 2 tasks
2019-03-25 19:35:44 INFO TaskSetManager:54 - Starting task 0.0 in stage 0.0 (TID 0, 192.168.0.112, executor 0, partition 0, PROCESS_LOCAL, 7870 bytes)
2019-03-25 19:35:44 INFO TaskSetManager:54 - Starting task 1.0 in stage 0.0 (TID 1, 192.168.0.112, executor 0, partition 1, PROCESS_LOCAL, 7870 bytes)
2019-03-25 19:35:44 INFO BlockManagerInfo:54 - Added broadcast_0_piece0 in memory on 192.168.0.112:53757 (size: 54.5 KB, free: 366.2 MB)
2019-03-25 19:35:45 INFO TaskSetManager:54 - Finished task 0.0 in stage 0.0 (TID 0) in 1207 ms on 192.168.0.112 (executor 0) (1/2)
2019-03-25 19:35:45 INFO TaskSetManager:54 - Finished task 1.0 in stage 0.0 (TID 1) in 1193 ms on 192.168.0.112 (executor 0) (2/2)
2019-03-25 19:35:45 INFO TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks have all completed, from pool
2019-03-25 19:35:45 INFO DAGScheduler:54 - ResultStage 0 (save at NativeMethodAccessorImpl.java:0) finished in 1.441 s
2019-03-25 19:35:45 INFO DAGScheduler:54 - Job 0 finished: save at NativeMethodAccessorImpl.java:0, took 1.497010 s
2019-03-25 19:35:45 INFO FileFormatWriter:54 - Write Job f4b0a10b-e9ec-4e47-ad2b-2b7e017cc41f committed.
2019-03-25 19:35:45 INFO FileFormatWriter:54 - Finished processing stats for write job f4b0a10b-e9ec-4e47-ad2b-2b7e017cc41f.

2019-03-25 19:35:45 ERROR __main__:? - End

2019-03-25 19:35:45 INFO AbstractConnector:318 - Stopped Spark@364fe62b{HTTP/1.1,[http/
{0.0.0.0:4041}

Another Example - Works Fine

```
def counting(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Count") \
        .getOrCreate()
    log4jLogger = spark.sparkContext._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger("count")
    counter = 0
    def count(item):
        global counter
        counter = counter + 1

    LOGGER.info("Start")
    df = spark.range(15)
    df.foreach(count)
    spark.stop()
```

Another Example - Works Fine

```
def counting(file):
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Count") \
        .getOrCreate()
    log4jLogger = spark.sparkContext._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger("count")

    counter = 0
def count(item):
    global counter
    LOGGER("In count")
    counter = counter + 1

LOGGER.info("Start")
df = spark.range(15)
df.foreach(count)
spark.stop()
```

Produces Error

Error

File "/Java/spark-2.4.0-bin-hadoop2.7/python/lib/pyspark.zip/pyspark/serializers.py", line 597, in dumps

_pickle.PicklingError: Could not serialize object: Py4JError: An error occurred while calling o26.__getstate__. Trace:

py4j.Py4JException: Method __getstate__([]) does not exist

at py4j.reflection.ReflectionEngine.getMethod(ReflectionEngine.java:318)

at py4j.reflection.ReflectionEngine.getMethod(ReflectionEngine.java:326)

at py4j.Gateway.invoke(Gateway.java:274)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.GatewayConnection.run(GatewayConnection.java:238)

at java.lang.Thread.run(Thread.java:748)

Log4j

Log Levels

OFF (most specific, no logging)
FATAL (most specific, little data)
ERROR
WARN
INFO
DEBUG
TRACE (least specific, a lot of data)
ALL (least specific, all data)

Can specify level
- Per package
- Per class

Can determine log
- Format
- Location of output

Setting Level in Code - How in Python?

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.log4j.{Level, LogManager, Logger}

object SimpleApp {
  def main(args: Array[String]) {

    Logger.getLogger("org").setLevel(Level.ERROR)
    val log = LogManager.getRootLogger
    log.info("Start")
    println("cat in the hat")
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val rdd = sc.parallelize(List(1,2,3,4))
    rdd.saveAsTextFile("SimpleAppOutput2")
    log.info("End")
    sc.stop()
  }
}
```

Output

```
->spark-submit --master spark://air-6.local:7077 simpleappintell_2.11-0.1.jar
log4j:WARN No appenders could be found for logger (root).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
cat in the hat
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
17/11/05 12:04:37 INFO root: End
```

Again - Do you want to set log level in Code?

Can set level in config file

$SPARK_HOME/conf/log4j.properties.temple

By default Spark will look for $SPARK_HOME/conf/log4j.properties

But is not part of program

Quiet Log config

```
# Set everything to be logged to the console
log4j.rootCategory=INFO, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Set the default spark-shell log level to WARN. When running the spark-shell, the
# log level for this class is used to overwrite the root logger's log level, so that
# the user can have different defaults for the shell and regular Spark apps.
log4j.logger.org.apache.spark.repl.Main=WARN

# Settings to quiet third party logs that are too verbose
log4j.logger.org=WARN
log4j.logger.org.apache.parquet=ERROR
log4j.logger.parquet=ERROR
```

Master Logging vs Slave Logging

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.log4j.{Level, LogManager, PropertyConfigurator, Logger}

object SimpleApp {
  def main(args: Array[String]) {
    val log = LogManager.getRootLogger
    log.info("Start")
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val rdd = sc.parallelize(1 to 10)
    val stringRdd = rdd.map { value =>
      log.info(value)
      value.toString
    }
    log.info("End")
    sc.stop()
  }
}
```

Master

Slave

Error on Running
Log on serializable

Serializable Logger - How in Python

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.log4j.{LogManager, Logger}

object DistributedLogger extends Serializable {
  @transient lazy val log = Logger.getLogger(getClass.getName)
}
```

Main

```
object SimpleApp {
  def main(args: Array[String]) {
    val log = LogManager.getRootLogger
    log.info("Start")
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val rdd = sc.parallelize(1 to 10)
    val result = rdd.map { i =>
      DistributedLogger.log.warn("i = " + i)
      i + 10
    }
    result.saveAsTextFile("SimpleAppOutput")
    log.info("End")
    sc.stop()
  }
}
```

Running

->spark-submit target/scala-2.11/simpleappintell_2.11-0.1.jar

17/11/06 16:59:40 INFO root: Start

17/11/06 16:59:41 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

[Stage 0:> (0 + 0) / 8]17/11/06 16:59:44 WARN DistributedLogger$ i = 7

17/11/06 16:59:44 WARN DistributedLogger$: i = 8

17/11/06 16:59:44 WARN DistributedLogger$: i = 9

17/11/06 16:59:44 WARN DistributedLogger$: i = 6

17/11/06 16:59:44 WARN DistributedLogger$: i = 3

17/11/06 16:59:44 WARN DistributedLogger$: i = 4

17/11/06 16:59:44 WARN DistributedLogger$: i = 1

17/11/06 16:59:44 WARN DistributedLogger$: i = 5

17/11/06 16:59:44 WARN DistributedLogger$: i = 2

17/11/06 16:59:44 WARN DistributedLogger$: i = 10

17/11/06 16:59:44 INFO root: End

Run In Standalone Master/Slave One Machine

```
def counting():
        counter = 0
        def count(item):
                global counter
                counter = counter + 1          <-------------  line 5

        from pyspark.sql import SparkSession
        spark = SparkSession.builder \
                        .appName("Counter") \
                        .getOrCreate()

        df = spark.range(16)
        smaller = df.coalesce(4)
        smaller.foreach(count)
        print(counter)
 if __name__ == "__main__":
         counting()
```

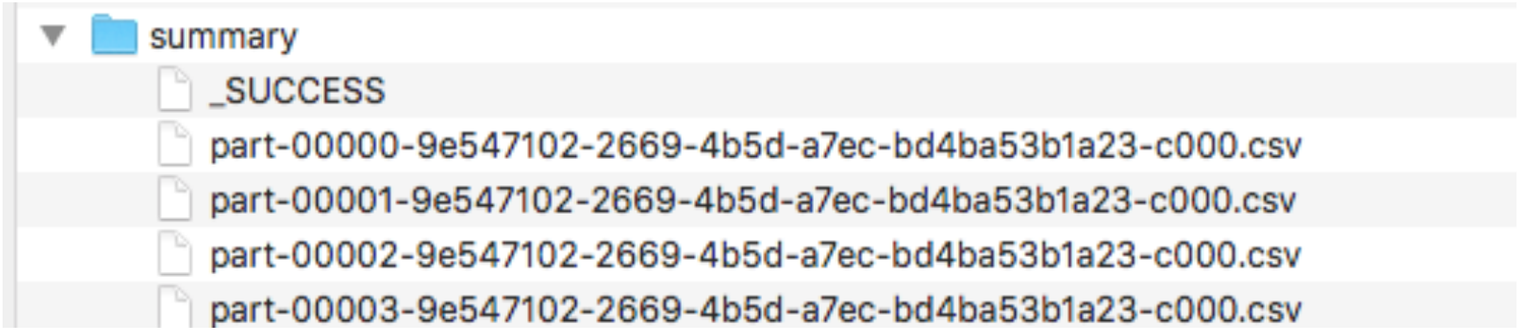
```
 File "/Users/whitney/Courses/696/Spring19/sparkExamples/./printExample.py", line 5, in count
    counter = counter + 1
NameError: name 'counter' is not defined
```

One More Example - Standalone, One Machine

```
def flight(input, output):
    import pyspark.sql.functions as F
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Fight") \
        .getOrCreate()

    flight_df = spark.read.json(input)
    grouped_df = flight_df.groupBy('DEST_COUNTRY_NAME').agg(F.sum('count'))
    fewer_parts = grouped_df.coalesce(4)
    fewer_parts.write.format('csv').save(output)

if __name__ == "__main__":
        inputfile, outputfile = files_from_args()
        flight(inputfile, outputfile)
```

Running & Output

spark-submit --master spark://rew-2.local:7077 ./flight.py -o summary -i ./2015-summary.json



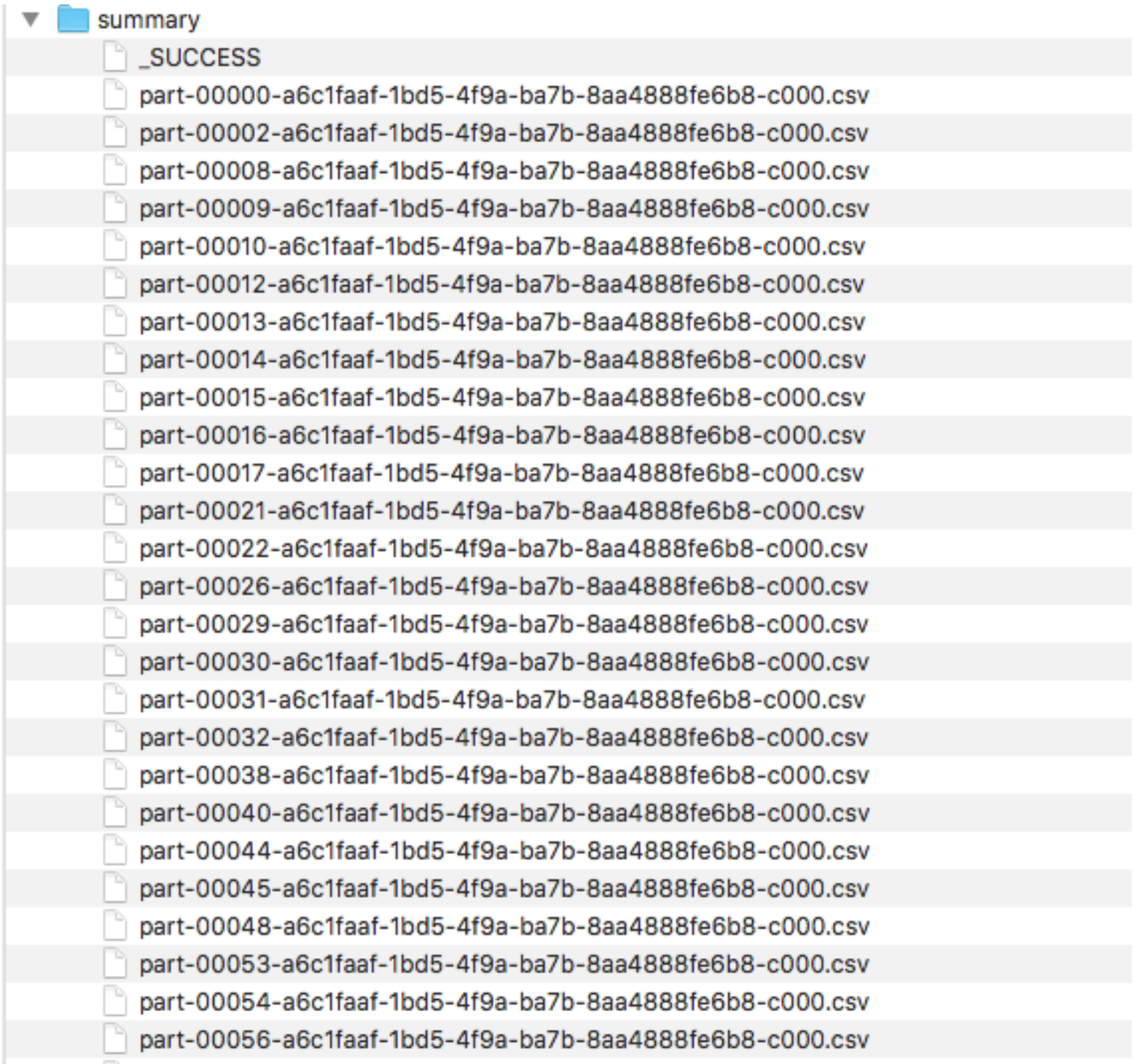
One More Example - Standalone, One Machine

No coalesce

```
def flight(input, output):
    import pyspark.sql.functions as F
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Fight") \
        .getOrCreate()

    flight_df = spark.read.json(input)

    grouped_df = flight_df.groupBy('DEST_COUNTRY_NAME').agg(F.sum('count'))
    grouped_df.write.format('csv').save(output)


if __name__ == "__main__":
        inputfile, outputfile = files_from_args()
        flight(inputfile, outputfile)
```

spark-submit --master spark://rew-2.local:7077 ./flight.py -o summary -i ./2015-summary.json

102 output files



Amazon Elastic Map-Reduce (EMR)

Hadoop, Hive, Spark, etc on Cluster

Predefined set of languages/tools available

Can create cluster of machines

https://aws.amazon.com

- Create new account
- Get 12 months free access

AWS Free Tier

I and students were charged last year

12 months free

EC2 - compute instances
- 740 hours per month
- Billed in hour increments
- Billed per instance

S3 - storage
- 5 GB
- 20,000 Get requests

RDS - MySQL, PostgresSQL, SQL Sever
- 20 GB
- 750 hours

EC2 Container - Docker images
- 500 MB

AWS Educate

https://aws.amazon.com/education/awseducate/

SDSU is an institutional member

Students get $100 credit

EC2 Pricing

	Price Per Hour	
	On Demand	Spot
m1.medium		$0.0047
m1.large		$0.0?
ml.xlarge		$0.352
m3.xlarge		$0.0551
m4.large	$0.1	$0.0299
c1.medium		$0.0132
c1.xlarge		$0.057

Basic Outline

Develop & test Spark locally

Upload program file & data to S3

Configure & launch cluster
- AWS Management Console
- AWS CLI
- SDKs

Monitor cluster

Make sure you terminate cluster when done

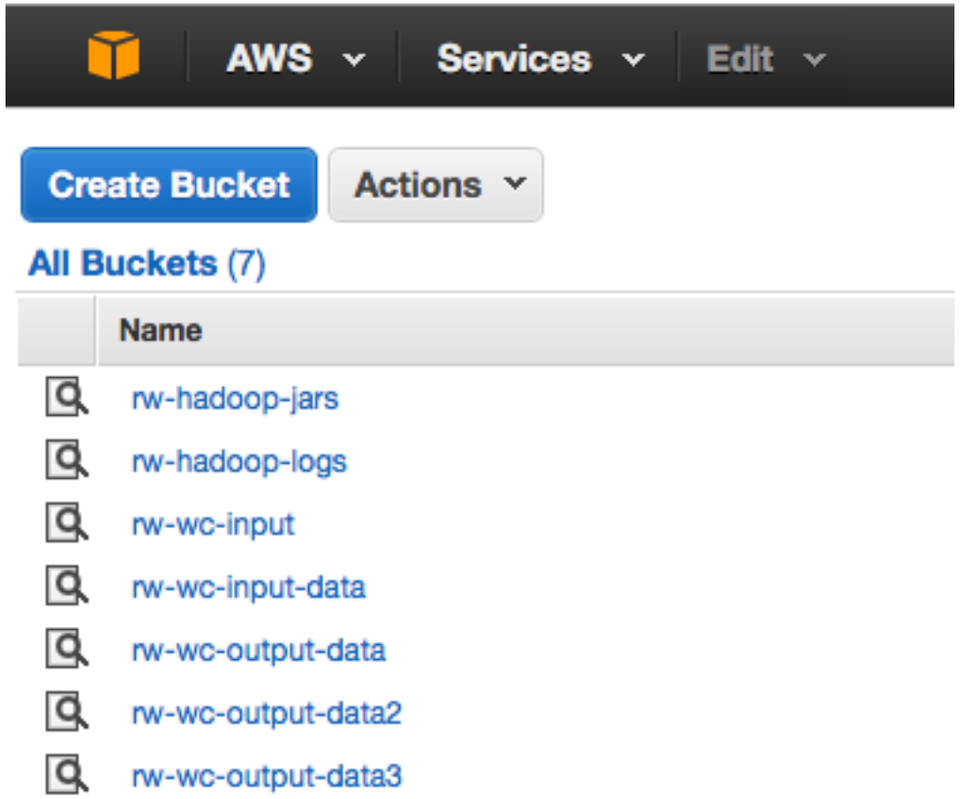
Simple Storage System - S3

Files are stored in buckets

Bucket names are global

Supports
- s3 - files divided in to block
- s3n

Accessing files
- S3 console
- Third party
- REST
- Java, C#, etc



Amazon S3

Amazon S3

Discover the new console Quick tips

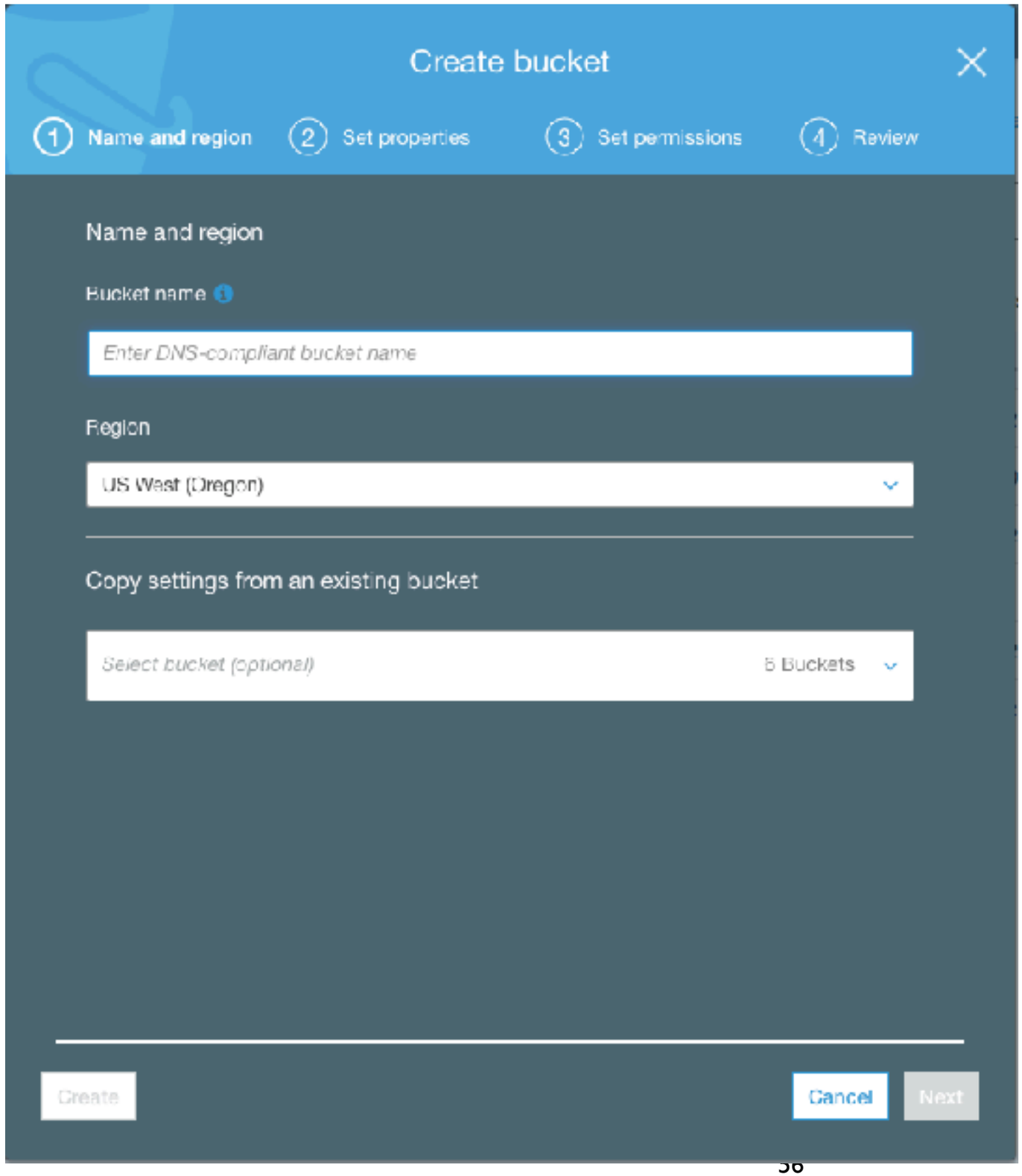
Search for buckets

+ Create bucket | Delete bucket | Empty bucket

6 Buckets 1 Regions

Bucket name	Region	Date created
aws-logs-834365227482-us-west-2	US West (Oregon)	Dec 8, 2016 3:27:27 PM
rw-hadoop-jars	US West (Oregon)	Nov 7, 2016 6:39:03 PM
rw-hadoop-logs	US West (Oregon)	Nov 7, 2016 6:51:29 PM
rw-output-data	US West (Oregon)	Nov 5, 2017 7:29:12 PM
rw-wc-input-data	US West (Oregon)	Nov 7, 2016 6:32:22 PM
rw-wc-output-data	US West (Oregon)	Nov 16, 2016 8:35:53 PM

S3 Creating a Bucket



S3 Costs

AWS Free Usage Tier

New AWS customers receive each month for one year
5 GB of Amazon S3 storage in the Standard Storage class,
20,000 Get Requests,
2,000 Put Requests, and
15 GB of data transfer out

	Standard Storage	Standard - Infrequent Access Storage	Glacier Storage
First 50 TB / month	$0.023 per GB	$0.0125 per GB	$0.004 per GB
Next 450 TB / month	$0.022 per GB	$0.0125 per GB	$0.004 per GB
Over 500 TB / month	$0.021 per GB	$0.0125 per GB	$0.004 per GB

S3 Objects

Objects contain
- Object data
- Metadata

Size
- 1 byte to 5 gigabytes per object

Object data
- Just bytes
- No meaning associated with bytes

Metadata
- Name-value pairs to describe the object
- Some http headers used
  - Content-Type

S3 Buckets

Namespace for objects

No limitation on number of object per bucket

Only 100 buckets per account

Each bucket has a name

Up to 255 bytes long

Cannot be same as existing bucket name by any S3 user

Bucket Names

Bucket names must

- Contain lowercase letters, numbers, periods (.), underscores (_), and dashes (-)
- Start with a number or letter
- Be between 3 and 255 characters long
- Not be in an IP address style (e.g., "192.168.5.4")

To conform with DNS requirements, Amazon recommends

- Bucket names should not contain underscores (_)
- Bucket names should be between 3 and 63 characters long
- Bucket names should not end with a dash
- Bucket names cannot contain dashes next to periods (e.g., "my-.bucket.com" and "my.-bucket" are invalid

Key

Unique identifier for an object within a bucket

Object Url

http://buckerName.s3.amazonaws.com/Key

http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl

Bucket = doc

Key = 2006-03-01/AmazonS3.wsdl

Access Control Lists (ACL)

Each Bucket has an ACL

- Determines who has read/write access

Each Object can have an ACL

- Determines who has read/write access

ACL consists of a list of grants

Grant contains

- One grantee
- One permission

S3 Data Consistency Model

Updates to a single object at a key in a bucket are atomic

But a read after a write may return the old value
  Changes may take time to progate

No object locking
  If two writes to same object occur at the same time
  The one with later timestamp wins

CAP Theorem

CAP theorem says in a distributed system you can not have all three

- Consistency
- Availability
- tolerance to network Partitions

Consistency

	Machine 1	Machine 2
	A = 2	A = 2
Not Consistent	A = 2	A = 3

Partition

Machine 1

A = 2

Machine 2

A = 2

Partitioned

Machine 1 cannot talk to machine 2

A = 2

A = 2

But how does machine 1 tell the difference between no connection and a very slow connection or busy machine 2?

Latency

Latency

Time between making a request and getting a response

Distributed systems always have latency

In practice detect a partition by latency

When no response in a given time frame assume we are partitioned

Available

	Machine 1	Machine 2	
Client	A = 2	A = 2	
Client	A = 2	A = 2	Client can not access value of A

What does not available mean?

- No connection
- Slow connection
- What is the difference?

Some say high available - meaning low latency

In practice available and latency are related

Consistency over Latency

Machine 1

Machine 2

Set A to 3

A = 2

A = 2

Set A to 3

A = 2

Lock A

A = 2

Write requests queued until unlocked

Set A to 3

A = 2

Set A to 3

A = 2

Increased latency

System still available

Set A to 3

A = 3

Unlock A

A = 3

A = 3

A = 3

Latency over Consistency

Machine 1

Machine 2

Set A to 3

A = 2

A = 2

A = 3

A = 2

Write requests accepted

Set A to 3

Low latency

System inconsistent

A = 3

A = 2

A = 3

A = 3

Latency over Consistency - Write Conflicts

Machine 1

Machine 2

Set A to 3

A = 2

A = 2

Subtract 1 from A

A = 3

A = 2

Set A to 3

A = 3

A = 1

Subtract 1 from A

A = ?

A = ?

Need policy to make system consistent

Partition

Machine 1

Machine 2

A = 2

A = 2

A = 2

A = 2

Set A to 3

A = 3

A = 1

Subtract 1 from A

A = ?

A = ?

Need policy to make system consistent

CAP Theorem

Not a theorem

Too simplistic
- What is availability
- What is a partition of the network

Misleading

Intent of CAP was to focus designers attention on the tradeoffs in distributed systems

- How to handle partitions in the network
- Consistency
- Latency
- Availability

CAP & S3

S3 favors latency over consistency