Introduction to Big Data with Apache Spark

This Workshop

Programming Spark

Resilient Distributed Datasets (RDDs)

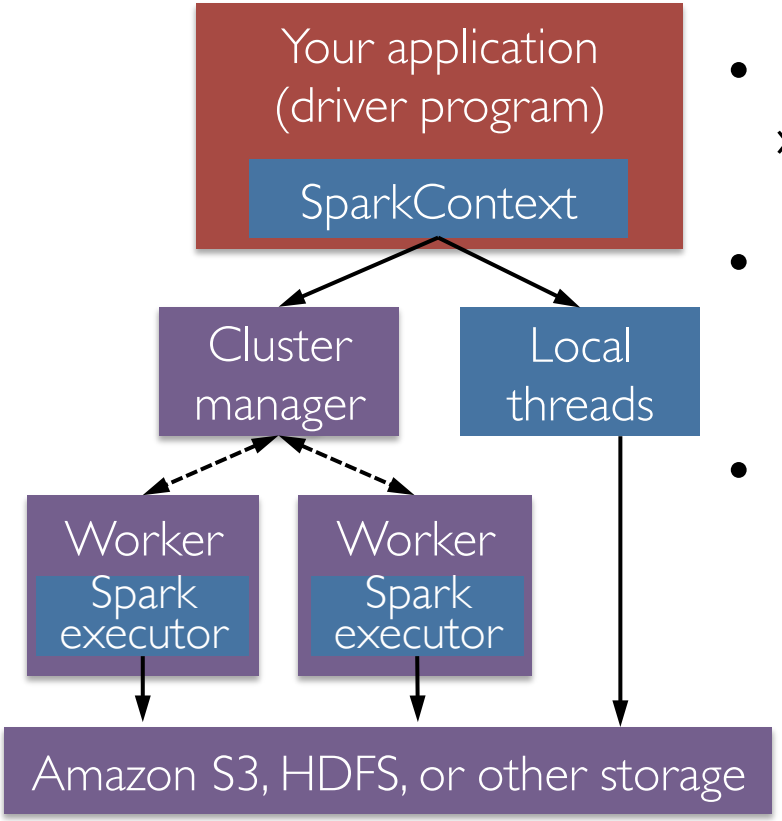
Creating an RDD

Spark Transformations and Actions

Python Spark (pySpark)

- We are using the Python programming interface to Spark (pySpark)
- pySpark provides an easy-to-use programming abstraction and parallel runtime:
  - » "Here's an operation, run it on all of the data"
- RDDs are the key concept

Spark Driver and Workers



- A Spark program is two programs:
  - » A driver program and a workers program
- Worker programs run on cluster nodes or in local threads
- RDDs are distributed across workers

Spark Context

- A Spark program first creates a **`SparkContext`** object
  - » Tells Spark how and where to access a cluster
  - » pySpark shell and Databricks Cloud automatically create the **`sc`** variable
  - » iPython and programs must use a constructor to create a new **`SparkContext`**
- Use **`SparkContext`** to create RDDs

Resilient Distributed Datasets

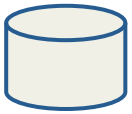
- The primary abstraction in Spark
  - » Immutable once constructed
  - » Track lineage information to efficiently recompute lost data
  - » Enable operations on collection of elements in parallel

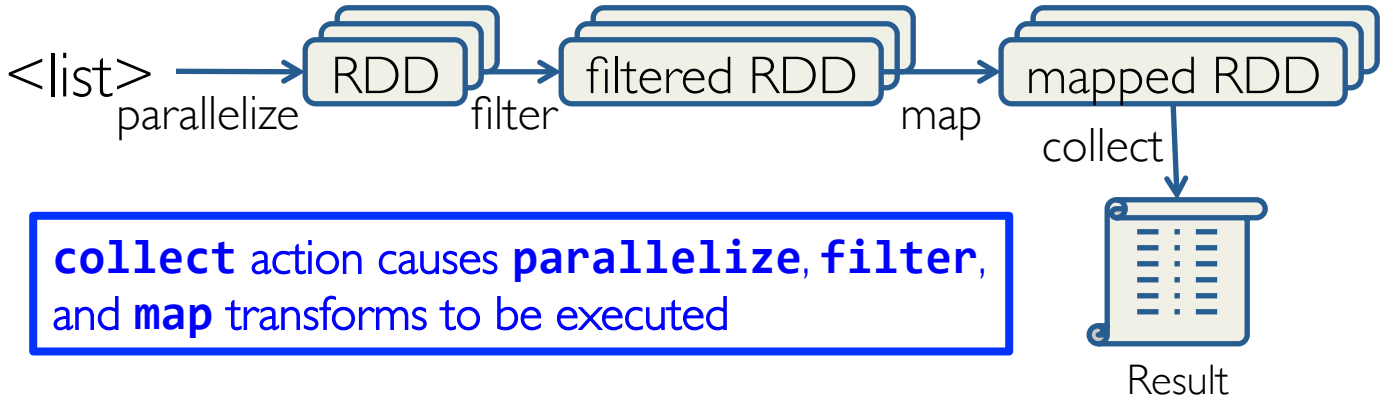
- You construct RDDs
  - » by *parallelizing* existing Python collections (lists)
  - » by *transforming* an existing RDDs
  - » from *files* in HDFS or any other storage system

RDDs

- Two types of operations: *transformations* and *actions*
- Transformations are lazy (*not computed immediately*)
- Transformed RDD is executed when action runs on it
- Persist (cache) RDDs in memory or disk

Working with RDDs

- Create an RDD from a data source:  <list>
- Apply transformations to an RDD: map filter
- Apply actions to an RDD: collect count



`collect` action causes **`parallelize`**, **`filter`**, and **`map`** transforms to be executed

Creating an RDD

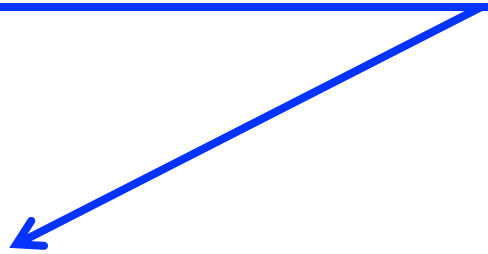
- Create RDDs from Python collections (lists)

```
>>> data = [1, 2, 3, 4, 5]
>>> data
[1, 2, 3, 4, 5]


>>> rDD = sc.parallelize(data, 4)
>>> rDD
ParallelCollectionRDD[0] at parallelize at PythonRDD.scala:229
```

No computation occurs with **`sc.parallelize()`**

- Spark only records how to create the RDD with four partitions



Creating RDDs

- From HDFS, text files, Hypertable, Amazon S3, Apache Hbase, SequenceFiles, any other Hadoop **`InputFormat`**, and directory or glob wildcard: **`/data/201404*`**

```
>>> distFile = sc.textFile("/user/hadoop/README.md", 4)

>>> distFile

/user/hadoop/README.md
MapPartitionsRDD[1] at textFile at

NativeMethodAccessorImpl.java:0
```

Spark Transformations

- Create new datasets from an existing one
- Use *lazy evaluation*: results not computed right away – instead Spark remembers set of transformations applied to base dataset
  - » Spark optimizes the required calculations
  - » Spark recovers from failures and slow workers
- Think of this as a recipe for creating result

Some Transformations

Transformation	Description
`map(`*`func`*`)`	return a new distributed dataset formed by passing each element of the source through a function *func*
`filter(`*`func`*`)`	return a new dataset formed by selecting those elements of the source on which *func* returns true
`distinct([`*`numTasks`*`]))`	return a new dataset that contains the distinct elements of the source dataset
`flatMap(`*`func`*`)`	similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item)

Review: Python `lambda` Functions

- Small anonymous functions (not bound to a name)
  `lambda a, b: a + b`
  » returns the sum of its two arguments
- Can use lambda functions wherever function objects are required
- Restricted to a single expression

Transformations

```
>>> rdd = sc.parallelize([1, 2, 3, 4])
>>> rdd.map(lambda x: x * 2)
RDD: [1, 2, 3, 4] → [2, 4, 6, 8]

>>> rdd.filter(lambda x: x % 2 == 0)
RDD: [1, 2, 3, 4] → [2, 4]

>>> rdd2 = sc.parallelize([1, 4, 2, 2, 3])
>>> rdd2.distinct()
RDD: [1, 4, 2, 2, 3] → [1, 4, 2, 3]
```

Function literals (green) are closures automatically passed to workers

Spark Actions

- Cause Spark to execute recipe to transform source
- Mechanism for getting results out of Spark

Some Actions

Action	Description
`reduce(`*`func`*`)`	aggregate dataset's elements using function *func*. *func* takes two arguments and returns one, and is commutative and associative so that it can be computed correctly in parallel
`take(`*`n`*`)`	return an array with the first *n* elements
`collect()`	return all the elements as an array WARNING: make sure will fit in driver program
`takeOrdered(`*`n, key=func`*`)`	return n elements ordered in ascending order or as specified by the optional key function

Getting Data Out of RDDs

```
>>> rdd = sc.parallelize([1, 2, 3])
>>> rdd.reduce(lambda a, b: a * b)
Value: 6

>>> rdd.take(2)
Value: [1,2] # as list

>>> rdd.collect()
Value: [1,2,3] # as list
```

Spark Program Lifecycle

1. Create RDDs from external data or parallelize a collection in your driver program
2. Lazily transform them into new RDDs
3. **`cache()`** some RDDs for reuse
4. Perform actions to execute parallel computation and produce results

Spark Key-Value RDDs

- Similar to Map Reduce, Spark supports Key-Value pairs
- Each element of a Pair RDD is a pair tuple

```
>>> rdd = sc.parallelize([(1, 2), (3, 4)])
RDD: [(1, 2), (3, 4)]
```

Some Key-Value Transformations

Key-Value Transformation	Description
`reduceByKey(`*`func`*`)`	return a new distributed dataset of (K,V) pairs where the values for each key are aggregated using the given reduce function *func*, which must be of type (V,V) ➔ V
`sortByKey()`	return a new dataset (K,V) pairs sorted by keys in ascending order
`groupByKey()`	return a new dataset of (K, Iterable<V>) pairs

Key-Value Transformations

```
>>> rdd = sc.parallelize([(1,2), (3,4), (3,6)])
>>> rdd.reduceByKey(lambda a, b: a + b)
RDD: [(1,2), (3,4), (3,6)] → [(1,2), (3,10)]

>>> rdd2 = sc.parallelize([(1,'a'), (2,'c'), (1,'b')])
>>> rdd2.sortByKey()
RDD: [(1,'a'), (2,'c'), (1,'b')] →
                [(1,'a'), (1,'b'), (2,'c')]
```

Spark References

- http://spark.apache.org/docs/latest/programming-guide.html
- http://spark.apache.org/docs/latest/api/python/index.html