CS 378 – Big Data Programming

Lecture 25

More RDD Types

Review

- Assignment 11: Inverted index in Spark
- We'll use the same data (Assignment 3)
  - Remove punctuation
  - For each word, output a list of verses containing the word, in sorted order
- Questions?

More RDD Types

- RDD containing Doubles
  - `JavaDoubleRDD`
  - Has actions specific to this type
    - `mean(),variance(), histogram()`

- RDD containing key/value pairs
  - `JavaPairRDD`
  - Has many actions specific to this type

Converting Between RDD Types

- Starting with a `JavaRDD`
- Convert to `JavaDoubleRDD` with:
  - `mapToDouble()`
    - Define: `DoubleFunction<T>`
    - Equivalent to: `Function<T, double>`
  - `flatMapToDouble()`
    - Define: `DoubleFlatMapFunction<T>`
    - Equivalent to: `Function<T, Iterable<Double>>`

- Could we start with another RDD type?

Converting Between RDD Types

- Starting with a `JavaRDD`
- Convert to `JavaPairRDD` with:
  - `mapToPair()`
    - Define: `PairFunction<T>`
    - Equivalent to: `Function<T, Tuple2<K,V>>`
  - `flatMapToPair()`
    - Define: `PairFlatMapFunction<T>`
    - Equivalent to: `Function<T, Iterable<Tuple2<K,V>>>`

- Could we start with another RDD type?

Converting Between RDD Types

- Suppose we start with `JavaPairRDD`
- Could we convert it to `JavaRDD`?

- Why would we want to?

- How would we do it?

Pair RDD

- Pair RDD in Java:
  - `JavaPairRDD`
- We've already created these in WordCount
- Pair RDDs have transformations specific to Pair RDDs
  - The pair defines a key, and a value
- Example: `reduceByKey()`, versus `reduce()`

Pair RDD Transformations

- Reduce by key
  - Values with the same key are passed to a reduce function

  - Source RDD element type: `<K, V>`
  - Result RDD element type: `<K, V>`

  - Java function (class) type: `Function2<V, V, V>`
  - Java method: `T call(T t1, T t2)`

Pair RDD Transformations

- Group by key
  - Values with the same key are grouped together

  - RDD element type: `<K, V>`
  - Result RDD element type: `<K, Iterable<V>>`

Pair RDD Transformations

- Map values
  - Apply a function to each value of the RDD

  - Source RDD element type: `<K, V>`
  - Result RDD element type: `<K, U>`

  - Java function (class) type: `Function<V, U>`
  - Java method: `U call(V v)`

Pair RDD Transformations

- Flat map values
  - Apply a function to each value of the RDD, return an iterable of values
  - Source RDD element type: `<K, V>`
  - Result RDD element type: `<K, U>`
  - Java function (class) type: `Function<V, Iterable<U>>`
  - Java method: `Iterable<U> call(V v)`

Pair RDD Transformations

- Keys
  - Returns an RDD containing only the keys
  - RDD element type: `<K, V>`
  - Result RDD element type: `K`
  - Type of the returned RDD?

Pair RDD Transformations

- Values
  - Returns an RDD containing only the values
  - RDD element type: `<K, V>`
  - Result RDD element type: `V`
  - Type of the returned RDD?

Pair RDD Transformations

- Sort by key
  - Sort the RDD elements by key

  - Source RDD element type: `<K, V>`
  - Result RDD element type: `<K, V>`

  - Keys must implement `Comparable`. Why?

Transformations on two Pair RDDs

- Subtract by key
  - Remove elements with a key present in the other RDD

  - Source RDD element type: `<K, V>`
  - "Other" RDD element type: `<K, V>`
  - Result RDD element type: `<K, V>`

  - Also have subtract: key and value must match

Transformations on two Pair RDDs

- Join
  - Inner join between two RDDs

  - Source RDD element type: `<K, V>`
  - “Other” RDD element type: `<K, U>`
  - Result RDD element type: `<K, Tuple2<V, U>>`

  - What if a key is not unique in an RDD?

  - Also have: leftOuterJoin, rightOuterJoin, fullOuterJoin

Transformations on two Pair RDDs

- Cogroup
  - For each key in either RDD, return lists of values from each

  - Source RDD element type: `<K, V>`
  - “Other” RDD element type: `<K, U>`
  - Result RDD element type:

    `<K, Tuple2<Iterable<V>, Iterable<U>>>`

Pair RDD Actions

- Additional actions
  - `countByKey()`
    - Returns a map from RDD element to count (integer)
  - `collectAsMap()`
    - Returns a map of the keys and values
  - `lookup(`*`key`*`)`
    - Returns a list of values associated with *`key`*