CS 378 – Big Data Programming

Lecture 20

Spark Basics

Review

- Assignment 9 – Download and run Spark
  - Controlling logging output

Review - Spark RDDs

- Resilient Distributed Dataset
  - One RDD has one or more partitions
  - Partitions are distributed across machines
  - Rebuilt from base data on failure (versus replication)
  - Lazy evaluation – created/computed on demand

- RDD types offer various functions
  - map, reduce
  - groupBy, reduceByKey
  - joins (inner, leftOuterJoin, rightOuterJoin)
  - filter, sample

Review - Spark Programs

- A Spark program defines:
  - RDDs
    - Input from external sources
    - Produced by a transformation
  - Transformations
    - Produce a new RDD from the input RDD(s)
  - Actions
    - Compute something from the input RDD
      - Return non-RDD objects (e.g., a number)
    - Write an RDD to external storage

Spark Actions

- Transformations define data flow (data lineage)
- Actions on RDDs initiate computation

- Counting
  - `rdd.count()`
- Extract some elements
  - `rdd.take(10)`
- Get all elements
  - `rdd.collect()`
  - Careful, as this assembles the entire RDD. Filter first.

Spark Transformations

- Transformations take functions as arguments
  - This function defines the details of the transform

- Filter
  - Apply a function to each element of an RDD, return those elements that evaluate to true
  - Source RDD element type: `T`
  - Result RDD element type: `T`
  - Java function (class) type: `Function<T, Boolean>`
  - Java method: `Boolean call(T t)`

Spark Transformations

- Map
  - Apply a function to each element of an RDD, return the result of applying the function
  - Source RDD element type: `T`
  - Result RDD element type: `R`
  - Java function (class) type: `Function<T, R>`
  - Java method: `R call(T t)`

Spark Transformations

- Flat Map
  - Apply a function to each element of an RDD, return the result of applying the function (an Iterator)
  - Source RDD element type: `T`
  - Result RDD element type: `R`
  - Java function (class) type: `FlatMapFunction<T, R>`
  - Java method: `Iterator<R> call(T t)`

  - Spark then iterates through each iterable returned,
  - to create an RDD with element type `R`

Spark Transformations

- Map to Pair
  - Apply a function to each element of an RDD, return the result of applying the function (a key and a value)
  - Source RDD element type: `T`
  - Result RDD element type: `<K, V>`
  - Java function (class) type: `PairFunction<T, K, V>`
  - Java method: `Tuple2<K, V> call(T t)`

Spark Transformations

- Reduce by Key
  - Apply a function to pairs of values with the same key, return the result (key and a "reduced" value)
  - Source RDD element type: `<K, V>`
  - Result RDD (JavaPairRDD) element type: `<K, V>`
  - Java function (class) type: `Function2<V, V, V>`
  - Java method: `V call(V v1, V v2)`

Spark Code Example

- WordCount again
- Alternative implementations of WordCount

Assignment 10

- Inverted Index in Spark
- Data set example
  - Genesis:46:14 And the sons of Zebulun; Sered, and Elon, and Jahleel.
- Output for this “document” (one verse)
  - How many “outputs” for this verse?
  - What should the keys be?
  - What should the value(s) be?

Assignment 10

- Input record parsing
  - Document-ID: `book:chapter:verse`
  - Text string that is the verse
- Verse text parsing:
  - Remove punctuation, lowercase all words
- Output:
  - Word, list of `book:chapter:verse` (no duplicates)
  - Sort the `book:chapter:verse` references
  - Extra credit: output a file ordered by the number of verses referenced for a word (descending)