CS5560 Knowledge Discovery and Management

Spark MapReduce Programing

Problem Set (PS-2B)
6/12/2017

Class ID: B 02

Name: Revanth Chakilam

Spark MapReduce Programming – Calculate everyone's common friends for Facebook

Facebook has a list of friends (note that friends are a bi-directional thing on Facebook. If I'm your friend, you're mine). They also have lots of disk space and they serve hundreds of millions of requests everyday. They've decided to pre-compute calculations when they can to reduce the processing time of requests. One common processing request is the "You and Joe have 230 friends in common" feature. When you visit someone's profile, you see a list of friends that you have in common. We're going to use MapReduce so that we can calculate everyone's common friends once a day and store those results. Later on it's just a quick lookup. We've got lots of disk, it's cheap.

1) Draw a MapReduce diagram similar to the word count diagram below.
2) Sketch a MapReduce algorithm for the common Facebook friends (referring to the word count code below).
3) Sketch Spark Scala implementation (referring to the word count code below).

Example

Assume the friends are stored as Person->[List of Friends], our friends list is then:

A -> B C D
B -> A C D E
C -> A B D E
D -> A B C E
E -> B C D

The result after reduction is:

(A B) -> (C D)
(A C) -> (B D)
(A D) -> (B C)
(B C) -> (A D E)
(B D) -> (A C E)
(B E) -> (C D)
(C D) -> (A B E)
(C E) -> (B D)
(D E) -> (B C)

① Map - Reduce Diagram :

Input

A → B C D
B → A C D E
C → A B D E
D → A B C E
E → B C D

Splitting

A → B C D
B → A C D E
C → A B D E
D → A B C E
E → B C D

Mapping

(A,B) → B C D
(A,C) → B C D
(A,D) → B C D.

(B,A) → A C D E
(B,C) → A C D E
(B,D) → A C D E
(B,E) → A C D E

(C,A) → A B D E
(C,B) → A B D E
(C,D) → A B D E
(C,E) → A B D E

(D,A) → A B C E
(D,B) → A B C E
(D,C) → A B C E
(D,E) → A B C E

(E,B) → B C D
(E,C) → B C D
(E,D) → B C D

Sort / Shuffle

(A,B) → (B C D) (A C D E)
(A,C) → (B C D) (A B D E)
(A,D) → (B C D) (A B C E)

(B,C) → (A C D E) (A B D E)
(B,D) → (A C D E) (A B C E)
(B,E) → (A C D E) (B C D)

(C,D) → (A B D E) (A B C E)
(C,E) → (A B D E) (B C D)

(D,E) → (A B C E) (B C D)

Reduce

(A,B) → (C D)
(A,C) → (B D)
(A,D) → (B C)

(B,C) → (A D E)
(B,D) → (A C E)
(B,E) → (C D)

(C,D) → (A B E)
(C,E) → (B D)

(D,E) → (B,C)

Final Result

(A,B) → (C D)
(A,C) → (B D)
(A,D) → (B C)
(B,C) → (A D E)
(B,D) → (A C E)
(B,E) → (C D)
(C,D) → (A B E)
(C,E) → (B D)
(D,E) → (B,C)

② Map - Reduce algorithm:

// person will be the key.

// friend list corresponding to him is the value

Note: friend is the every other person from friends list in common

* class Mapper

```
map (person, friends list) {
for each person in the friends list {
    emit (<person, friend>, friends list)
    }}
```

* class Reducer

```
reduce (<person, friend>, friends list) {
for each person, friend in friends list {
    Common friends = common friends (person, friend)
emit (<person, friend>, common friends)
    }}

Common friends (person, friend) {
if (person < friend) {
    list.add (friend)}
    else {
        list.add (person)
        }
        return list
        }
```

③ Spark Scala Implementation :

```
Val friends_data = SparkContext.textFile("f.txt")
Val mapper = friends data.flatmap(line => {
  Val person = line.split("->")
    Val key = person(0)
  Val common_friends = person(1).split(" ").map(friend =>
  {
if (key < friend)(key, friend) else (friend, key) })
    Common_friends.map(pair => (pair, person(1).toArray))
    })
    Val list = common_friends.reduceByKey(_intersect_)
    .list.SaveAsTextFile("output.txt")
```