**CS5560 Knowledge Discovery and Management**

Spark MapReduce Programing

PS2-B

6/12/2017

Class ID: Sai Sriharsha Sudulaguntla

Name: 25

**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.

**For 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)

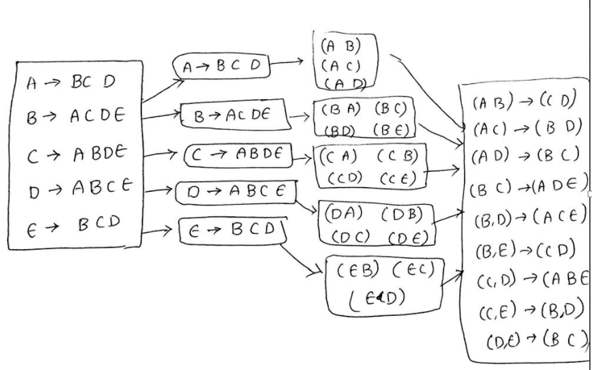
(C E) -> (B D)

(D E) -> (B C)

Now when D visits B's profile, we can quickly look up (B D) and see that they have three friends in common, (A C E).

1. Draw a MapReduce diagram similar to the word count diagram below.

Map Reduce:



1. Sketch a MapReduce algorithm for the common Facebook friends (referring to the word count code below).

The MapReduce solution to find ”common friends” has a map() and reduce()functions. The algorithm is as follows:

1. The mapper accepts a (key1, value1) pair, where key1 is a person and value1 is a list of associated friends of this person.

2 .The mapper emits a set of new (key2, value2) pairs where key is a Tuple2(key1, friend i) where friend i ∈ value1and value2 is the same as value1 (list of all friends for key1).

3. The reducer’s key is a pair of two users (User j, User k) and value is a list of sets of friends.

4. The reduce() function will intersect all sets of friends to find common and mutual friends for (User j, User k) pair.

1. Sketch Spark Scala implementation (referring to the word count code below).

**Map:**

Map(key, value)

{

reducerValue = (friend\_1, friend\_2… friend\_n);

foreach friend in (friend\_1, friend\_2… friend\_n)

reducerKey = buildSortedKey(person, friend);

emit(reducerKey, reducerValue);

}

**Reduce:**

Tuple2 buildSortedKey(person, friend)

{

if (person1, person2)

{

return Tuple2(person1, person2)

}

else (person1, person2)

{

return Tuple2(person2, person1)

}

}