

Lecture 8

MapReduce Examples

MapReduce examples,

Refer slide time :(0:15)

Example: 1 Word Count using MapReduce

map(key, value):

// key: document name; value: text of document

for each word w in value:

emit(w , 1)

reduce(key, values):

// key: a word; values: an iterator over counts

result = 0

for each count v in values:

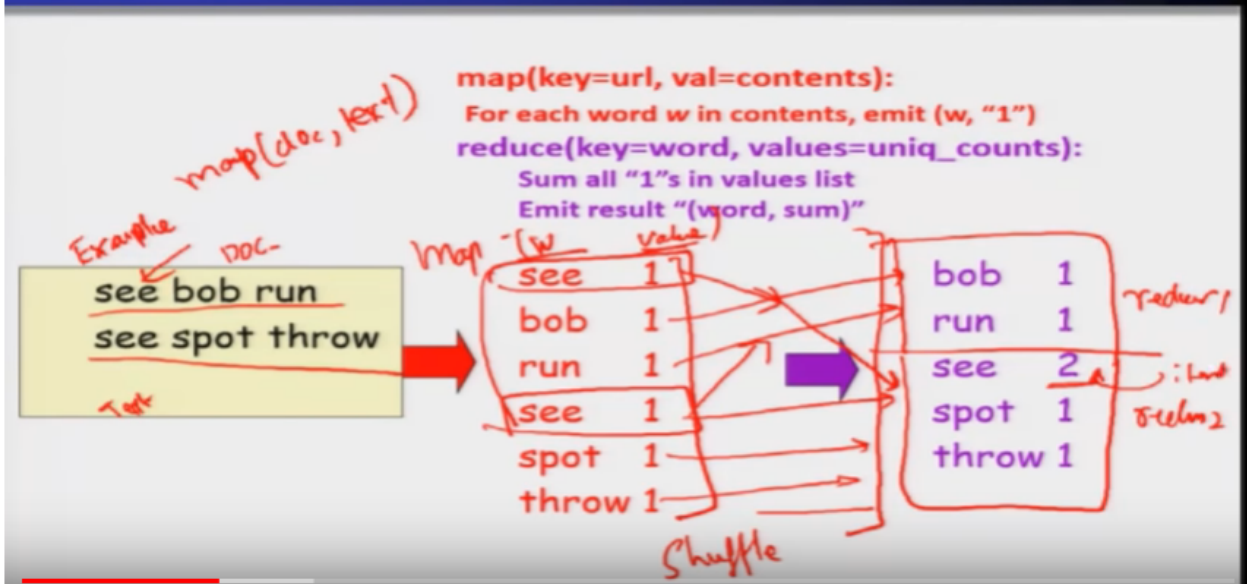
result += v

emit(key, result)

Example one, what count using MapReduce? So, here we see the structure of map and reduce program which, will do the word count, in a given document. So, the input we assume is a document with the name is given as the key and the text, of that document is the value for each of our W which appears in a particular text of a document, this map will omit that name of that word and the value 1. Now this particular, intermediate result, that is the word and this value, is taken as an input in the reduced function. So, reduce will get a key as a word which is emitted by the map function and the values as an iterator over the counts. So, the counts which is emitted by the map function, is the value and here also this word is, output by the might map amid. Now this iterator will run on this count values so, for example for each count value V , in the values it will do the sum and for every key it will result the output.

Refer slide time :(01:57)

Count Illustrated



Let us see this illustration through this running example, now in this example we consider a document, let us say this name of a document, docx file and this is the text, of the document now, when this particular name of the document, is given and the text is given. So, these particular text words, as it is appearing it will emit word and W. So, here we see that, W and this is the value 1 and this is the value. So, as these words are appearing out of this particular text, this map function will emit, these W and 1 out of this particular previous program. So, after this emit of these words, it will do a shuffle, in so-called, what it will do the same words it will try to, collect together, it will sort them and collect them, together and then pass on, this is the shuffle phase and then pass on to, the reducer phase. Now it depends upon how many reducers we have let us assume we have one and two different reducers. So, as far as this Bob is concerned, it will be passed on over here this will pass on over here Bob will go to this particular function a run will go to this particular function, to this reducer I spot will go here, here and throw will go here. And this reducer will now, for example in C, there are two different, these values when it comes and iterator it will, go through this iterator function and make the summation of it.

Refer slide time :(04:34)

Example 2: Counting words of different lengths

- The map function takes a value and outputs key:value pairs.
- For instance, if we define a map function that takes a string and outputs the length of the word as the key and the word itself as the value then
- map(steve) would return 5:steve and
- map(savannah) would return 8:savannah.

Example Doc.
This is my pen
4 2 2 3
(4:1) (2:2) (3:1)

This allows us to run the map function against values in parallel and provides a huge advantage.

Now let us see another example, example number two, counting words of different lengths. So, in a given document we have to find out, the documents of a given length for example, this is my pen. So, here you see that this word is of length four this word is of length tow this word or is of length tow this word is of length three. So, counting the words of different lengths is, you see this particular word, of length 2 is appearing two times, the word of length four is appearing one times word of length three is appearing one times this we have to give as an output from for a given document file. So, let us see how map and reduced function, is utilized to do this particular job.

Refer slide time :(05:40)

Example 2: Counting words of different lengths

Before we get to the reduce function, the mapreduce framework groups all of the values together by key, so if the map functions output the following **key:value pairs**:

3 : the
3 : and
3 : you
4 : then
4 : what
4 : when
5 : steve
5 : where
8 : savannah
8 : research

map
emit (length, word)
↑
key *value*
reduce
(key, list)

They get grouped as:

3 : [the, and, you] ✓
4 : [then, what, when] ✓
5 : [steve, where] ✓
8 : [savannah, research] ✓

Now to do this counting words of different lengths, we have to design the map and reduce function. So, given this particular document, the map function will emit the key and the value. So, the key comprises of the length, of about and the word itself. So, for example, at ERT the this is the word, this will be emitted and also the length, will be emitted so key value pair will be key, will be the length, here length of the world and the world itself will become the value this will be emitted out of the map function. Now after that what the reducer? Will do reducer will accept this format and then it will be group by so, the reducer, will now collect the, the key and corresponding it will outcome the list of the words. So, for example here, you can see the word of length 3 it is 3 different types of 3 different words, which are appearing, is being clubbed together, as a list. So, this will be the output of the reduce function.

Refer slide time :(07:27)

Example 2: Counting words of different lengths

- The reductions can also be done in parallel, again providing a huge advantage. We can then look at these final results and see that there were only two words of length 5 in the corpus, etc...
- **The most common example of mapreduce is for counting the number of times words occur in a corpus.**

So, the reductions can be done in parallel, again providing a huge advantage, we can then look at these final designs and see that there are only two words of length 5 in the corpus and only two words of length 8 and so on. So, this we have seen through an example that using MapReduce program complex program or application can be easily programmed and this is going to be used in a big data applications.

Refer slide time :(08:01)

Example 3: Word Length Histogram

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled. When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change. We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

How many "big", "medium" and "small" words, are used ?

Example number 3; Here we have to find out the word length histogram and if this particular document is given and then we have to find out that how many big medium and small words are appearing in this particular document and this becomes the word length histogram.

Refer slide time :(08:23)

Abridged Declaration of Independence

- [illegible]

Refer slide time :(09:19)

Example 3: Word Length Histogram

Split the document into chunks and process each chunk on a different computer

Chunk 1

Chunk 2

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

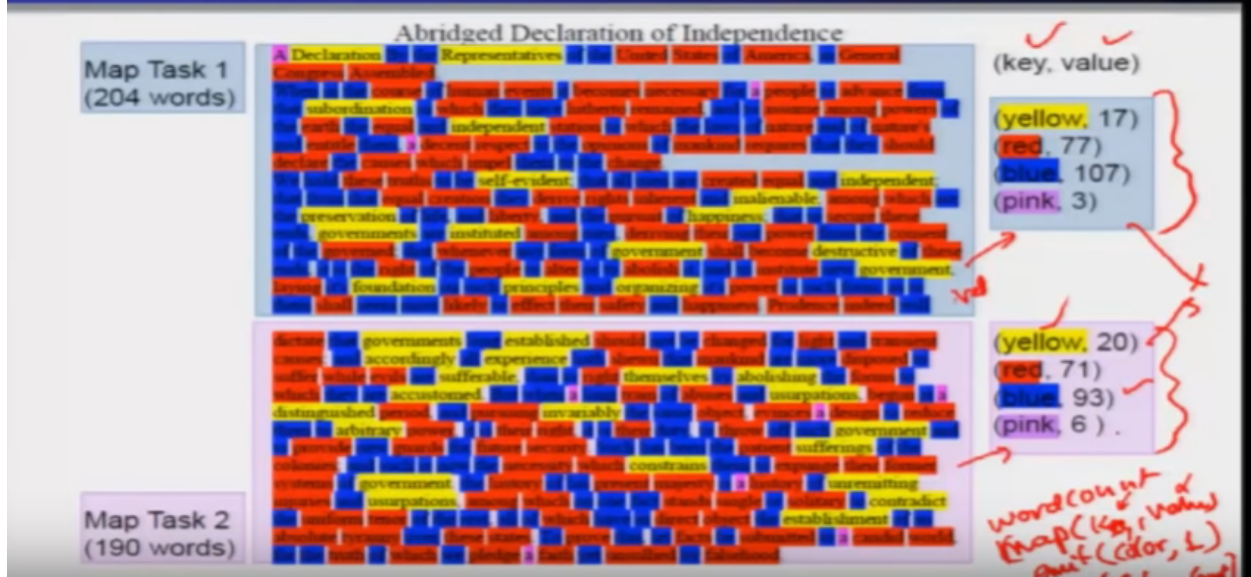
We hold these truths to be self-evident, that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying its foundation on such principles and organizing its power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government, the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unshaken by falsehood.

Now to do this you know that in MapReduce program, we can divide this entire document into the chunks let us assume that this is upper, when this particular this portion of a document, is chunk one the other portion the remaining portion is chunk so, we divided into two chunks.

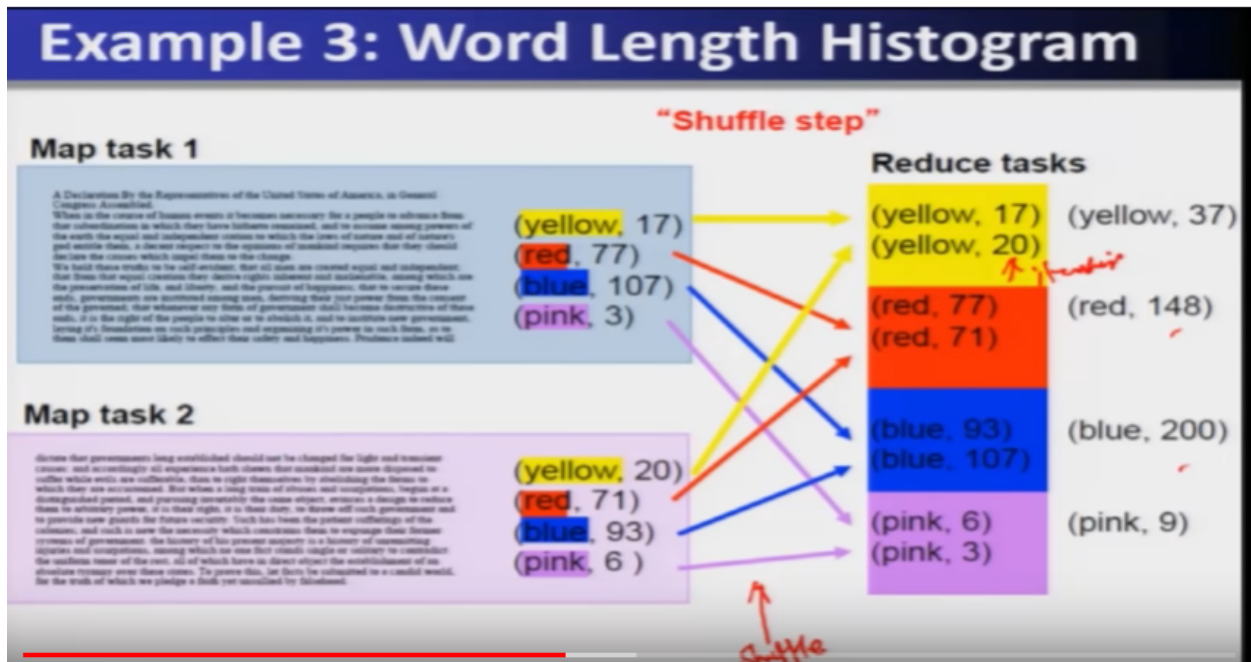
Refer slide time :(09:41)

Example 3: Word Length Histogram



And now we have to write down the map function. So, map function will be based on, this word count application, wherein if a word is given and the map functions will take the key and the value. So, key will be this document and this value will be these words out of this particular, document is written now whenever a word is, there then through this particular length, we have to find out whether it, will emit whether it is the color, of our award and one will be emitted, similarly as far as the reduced function, is concerned reduce for every key it will make a sum of a count, I it will do the aggregation. So, count means the iterator, iterator will make the sum and this will be the output. So, in this the, the chunk one will omit this statistics that is yellow 17 different times it is a appearing red 77 blue 107 and pink is 3 similarly the, the map task 2, will emit the yellow as 20 red as 71 and blue as 93. So, these numbers are internally done, that means that every task the reduce function is applied that is why instead of 17 times once it is doing this, now this reduce again will combine them and give the final outcome that yellow is 37 and red is 148 and blue is 1200 and this pink is 9.

Refer slide time :(11:56)



Reduce tasks

(yellow, 17) (yellow, 37)
(yellow, 20) (yellow, 37)
(red, 77) (red, 148)
(red, 71) (red, 148)
(blue, 93) (blue, 200)
(blue, 107) (blue, 200)
(pink, 6) (pink, 9)
(pink, 3) (pink, 9)

So, if we see these different steps so, this step is called a, 'Shuffle Step'. And through this different 17 and 20 here there will be an iterator and thus reduce function will long add, after going through this iterator and these values will outcome. So, this example shows the word length histogram which is nothing but an extension of word count program

Refer slide time :(12:28)

Example 4: Build an Inverted Index

Input:

tweet1, ("I love pancakes for breakfast")
 tweet2, ("I dislike pancakes")
 tweet3, ("What should I eat for breakfast?")
 tweet4, ("I love to eat")

Desired output:

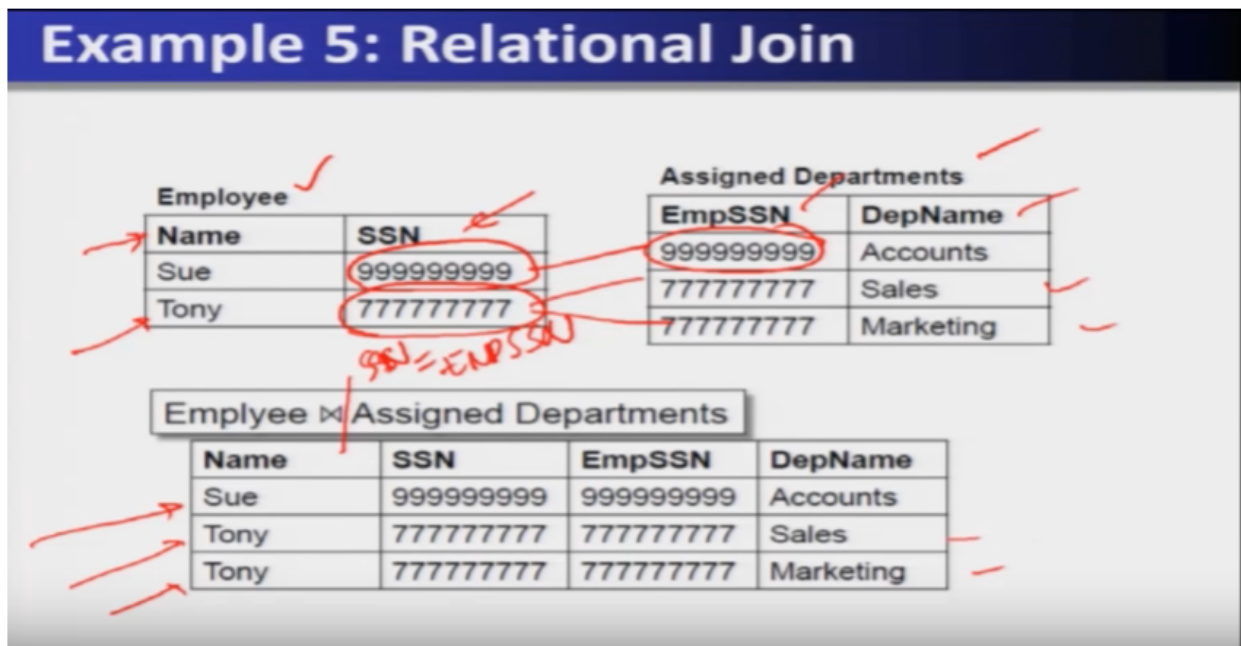
"pancakes", (tweet1, tweet2)
 "breakfast", (tweet1, tweet3)
 "eat", (tweet3, tweet4)
 "love", (tweet1, tweet4)
 ...

Handwritten notes:

- Set Document* (pointing to the input tweets)
- map(Key, Value)* (in a box)
- emit(Word, Doc-ID)* (in a box)
- Ex - (Pancakes, tweet1)*
- (breakfast, tweet1)*
- (Pancakes, tweet2)*
- (breakfast, tweet3)*
- (eat, tweet3)*
- (love, tweet4)*
- reduce & emit (words, list doc-ID)*
- Pancakes, (tweet1, tweet2)*

Now there is another example for to build an inverted index. So, by mean inverted index is, for example if a document is given or a set of documents is, is given then we want to find out the, the text or a particular word, which is appearing in a particular document and then we will finally collect a world which is appearing in all list of all documents where it is appearing, this is called, 'Inverted Index'. And search engines are now using this concept. So, let us see how this happens so, map function will take a key value pair and will emit the, the name of the word that is called, 'Value'. What value? And the document ID it will emit. So, for example in this particular case, here the world like pancake pancakes, it will emit the document ID that is to each one and it will again emit, let us say a breakfast, this is to eat one similarly in the next one, is called, 'Pancakes', to eat two and then in the next one breakfast, tweet three and here eat 93 then low then tweet three. Now this is an intermediate, key value pairs which is emitted by the malfunction, now using this we have to find out the, inverted index that is a desired output in whatever index by mean that the reduced function, will group by keys and will list out all that particular document IDs where it is appearing. So, that reduce will basically emit the output as, as these words and the list of document IDs. So, therefore in this case, pancake if you see is appearing in this to each one, is appearing in this pancake is appearing in tweet one and also the pancake is appearing in to it too similarly the breakfast if you see, is appearing in to it one and the breakfast is also appearing in to it three, similarly it is appearing in tweet three and also in to it four. Now is appearing in to it one and to eat four. So, this will form the inverted index, using simple MapReduce program and whenever a search engine uses, the word breakfast, for searching so, it will give these two documents, to it one and two it three where this breakfast, was being referred in that document.

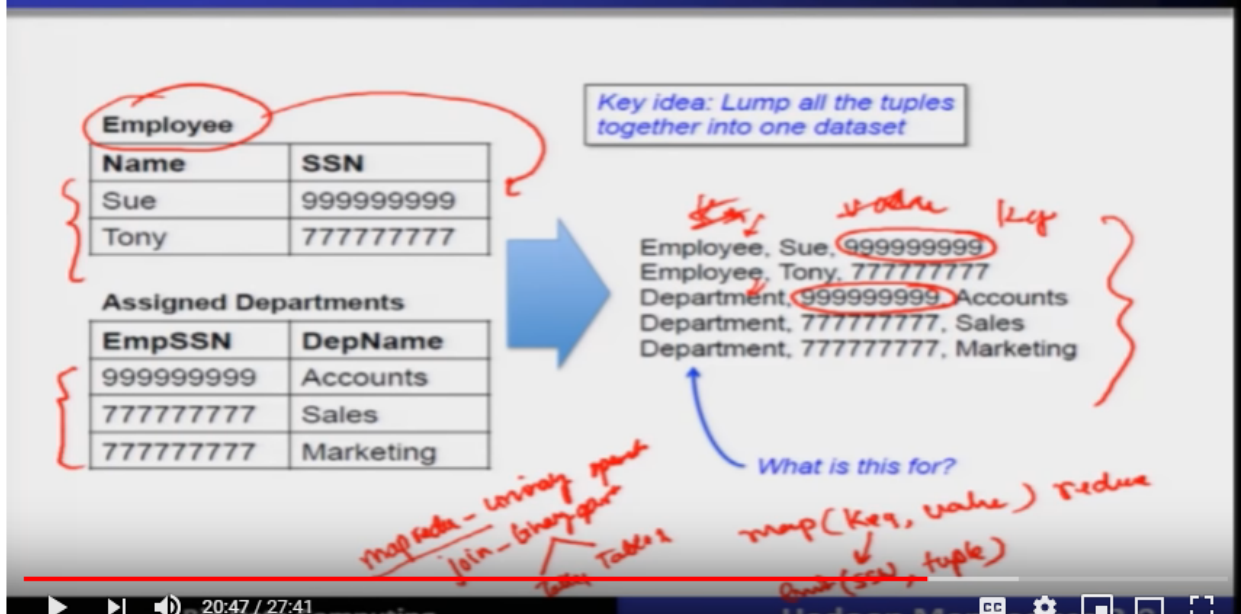
Refer slide time :(16:48)



Now we will see the operations, how we are going to perform the relational, join operation using MapReduce, by relational join let us understand this example and then we will see, how that is done using MapReduce. Let us say that employee is a table having, the attributes as a name and its SSN number, similarly another document which is called, 'Assigned Departments', here we have in SSN and the department name. Now if you want to join on, employee and assigned departments, if you want to join on SSN equal, employee SSN then we, we see that this particular employee with the name sue has matching, employee SSN with the accounts. So, if we join them this particular tuple will be generated, what about the other one here this SSN has two matches. So, therefore this particular tuple two different tuples will be generated accordingly wherein this is the sales and marketing will be reflected. Now let us see how this we are going to achieve using MapReduce operation.

Refer slide time :(18:25)

Example 5: Relational Join: Before Map Phase



Now before going into details we have to understand that map and MapReduce is a unary operation and this join is a binary operation, when it requires two tables, table 1 and table 2. So, how this MapReduce which is a unary operation, will be used to do a relational join and that we have to see that now what we will do is we consider the entries or the tuples of the table as, as a single tuples, as the collection of all the tuples and we will attach this identity of the name of a table also. So, it becomes a key value pair so, key value pair means that, the name of the table will become let us say key and the remaining couple will become the value this way we will list out all the tuples, which are there in different tables. Now if this becomes, a complete data set then we can perform the join operation, easily how the join is happening is join is happening around a particular key. So, around a SSN number so SSN number will become the key here in this case and we will omit the, the Map Reduce will, will emit the key and the value. So, key will be this SSN number and the value it will emit will be the entire tuple. So, now as far as the reduce is concerned reduced, will now group by this key and, and then after group by key then, it will try to do the iterator and if these table, IDs are different, then it will make a joint operation within it.

Refer slide time :(20:50)

Example 5: Relational Join: Map Phase

Employee, Sue, 999999999
Employee, Tony, 777777777
Department, 999999999, Accounts
Department, 777777777, Sales
Department, 777777777, Marketing



Join ✓

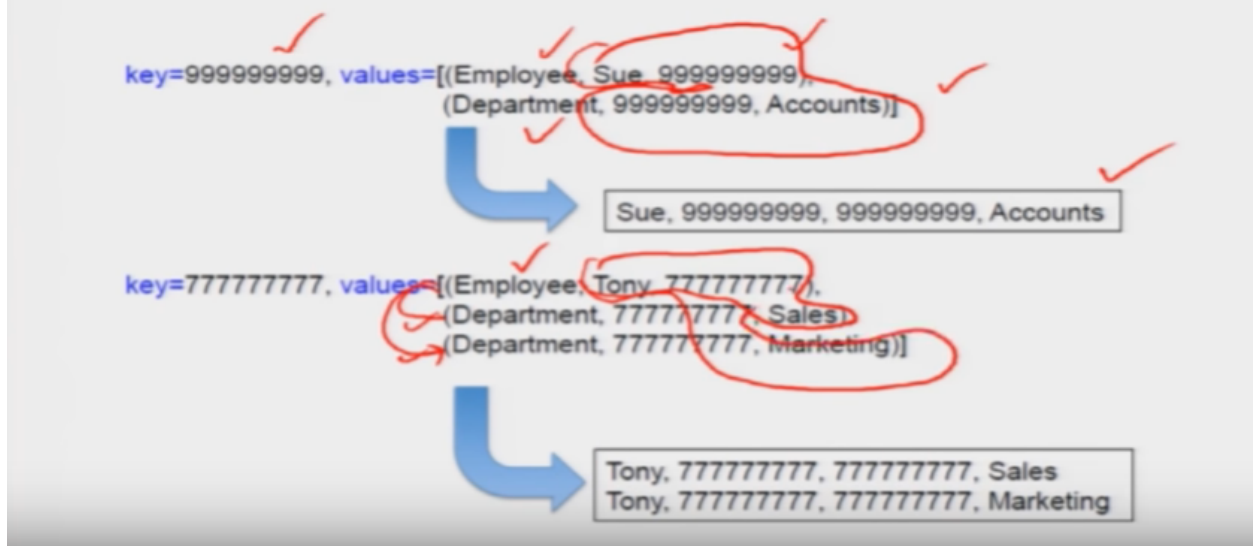
key=999999999, value=(Employee, Sue, 999999999)
key=777777777, value=(Employee, Tony, 777777777)
key=999999999, value=(Department, 999999999, Accounts)
key=777777777, value=(Department, 777777777, Sales)
key=777777777, value=(Department, 777777777, Marketing)

why do we use this as the key?

So, let us see here in this case as I told you that you we have to decide what is the key and what is the value. So, key will be the, the, the SSN number around which we are going to join. So, whatever is the joint become, the key and the entire tuple including the name of the table or a name of the document s and all the tuple will be that value. So, this after, after finding out this kind of

Refer slide time :(21:23)

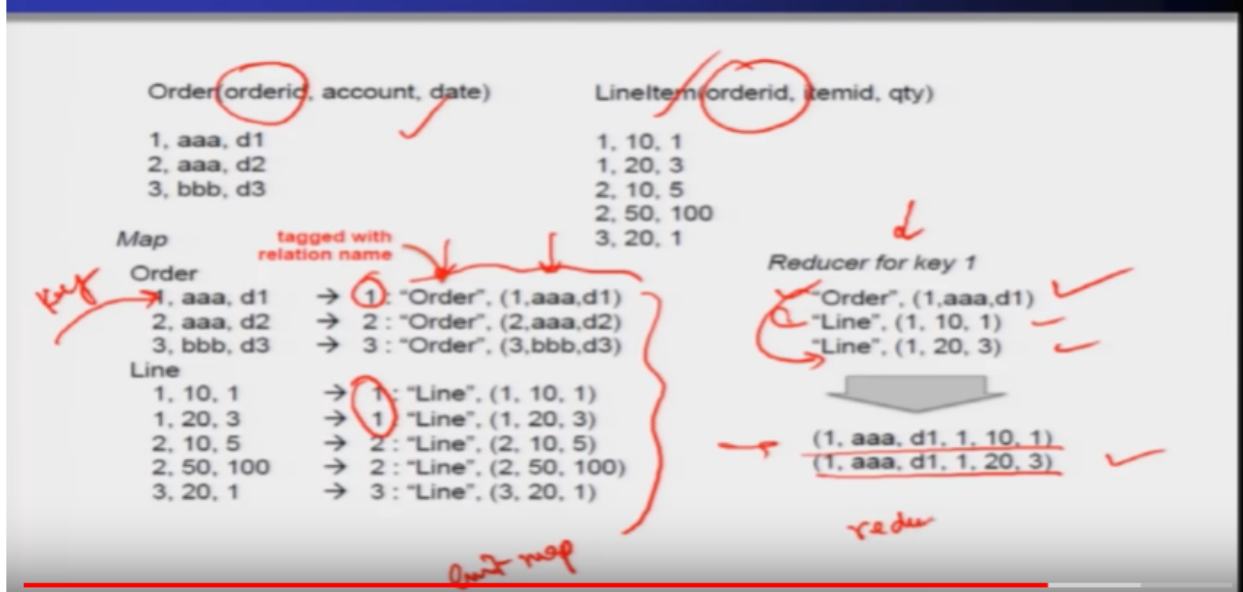
Example 5: Relational Join: Reduce Phase



things, then now we have to do a join operation. So, in the join we will see that when we make a group by key, let us say that two different tuples are appearing and since they are employ their, their tables are different. So, we are going to join them using these different notions and similarly here in the other case, we have a employ and two different Department. So, this employee will be joined so, it will generate two different tuples where in the Tony and sales, is one tuple Tony and marketing will be another tuple this way the, join operation.

Refer slide time :(22:10)

Example 5: Relational Join in MapReduce, again



Relational join we can perform now if you take another example, let us say that we have two different tables, one is the order table the other is line item table and their tuples are listed over here then with a map function, what we will do is so, we have to join around order ID, if we are going to join around order ID. So, order ID will become the key so, the map will emit the key and the corresponding all the values that is the name of the table and the, the corresponding. So, far both the tables it will generate this, this will emit out of the map function. Now then the reducer will combine, by this one key so, it will group by key for example this order, and for example key number one is appearing three times. So, one with the order the other is with the line. Now then we are going to combine this order with two different lines. So, basically it will generate two different tuples which is shown over here. so, by this example we have shown

Refer slide time :(23:32)

Example 6: Finding Friends

- 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. This list doesn't change frequently so it'd be wasteful to recalculate it every time you visited the profile (sure you could use a decent caching strategy, but then we wouldn't be able to continue writing about mapreduce for this problem).
- 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.

that using MapReduce, we can perform the relational join, example number six, for finding the common friends. So, this kind of finding friends is very common in the social networks, like, Facebook. So, Facebook has a list of friends and we want to find out the ur list of common friends, which are there in the social network like, our Facebook now this kind of operation of finding common friends is quite common, when you visit someone profile and see the list of friends that you have the common this list doesn't change frequently and but it has to be calculated, at a regular intervals using Facebook.

Refer slide time :(24:32)

Example 6: Finding Friends

- 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

input:
Person -> [List of friends]

So, let us see that how using MapReduce we are going to perform this kind of operation. So, assume that the friends are is told in this format, for example a person and the list of all the friends is basically given as the input for example person a, is having a list of friends, B C D. Similarly, B is having AC de and so on. So, the person and arrow followed by, the list of friends is given as the input to this particular program, now then we will see that.

Refer slide time :(25:11)

Example 6: Finding Friends

For map(A -> B C D) :

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

AB -> BCD
AC -> BCD
AD -> BCD

For map(B -> A C D E) : (Note that A comes before B in the key)

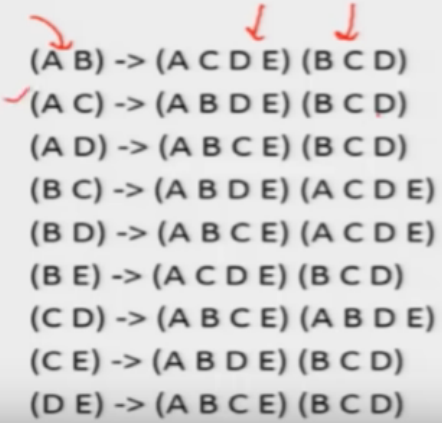
- (A B) -> A C D E
- (B C) -> A C D E
- (B D) -> A C D E
- (B E) -> A C D E

How to find out a common friend, is first thing is for a map function, what we will do is? So, if the person a and all the list of common friends is given so, we are going to generate a tuple that is of a B ,then a C then a D and we will output the same set of list, that is b c d b c d then b c d, similarly for another case also with B we have B a, then B C ,then B D and B E and all these same tuple will be generated, that is in the form of a key value pair ,by the map function.

Refer slide time :(26:00)

Example 6: Finding Friends

- Before we send these key-value pairs to the reducers, we group them by their keys and get:



```

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

And finally the next thing has to be done by the reducer. So reducer will find out, reducer will reduce a group by key, for example a B is a key and it has two different, such lists which has been obtained and now similarly, for a key a see two different lists and so. So, once this is obtained so, reducer

Refer slide time :(26:24)

Example 6: Finding Friends

- Each line will be passed as an argument to a reducer.
- The **reduce function will simply intersect the lists of values** and output the same key with the result of the intersection.
- For example, **reduce((A B) -> (A C D E) (B C D))**
will output **(A B) : (C D)**.
- **and means that friends A and B have C and D as common friends.**

Will find out by taking the intersection, of these lists of values and that becomes stuck. A list of common friends, for example, a B has two different lists, a C D E and B C D. So, if we take the intersection, which is common, is C D is appearing in both the lists, hence the list of common friends between a and B, is C D. Now this operation will be performed.

Refer slide time :(26:49)

Example 6: Finding Friends

- 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).

In this particular manner, by finding the intersection and for every set of persons, now in this manner, we are computing the list of common friends, in parallel and

Refer slide time :(26:04)

Reading

Jeffrey Dean and Sanjay Ghemawat,

“MapReduce: Simplified Data Processing on Large Clusters”

<http://labs.google.com/papers/mapreduce.html>

this particular operation is being performed. Thank you

