

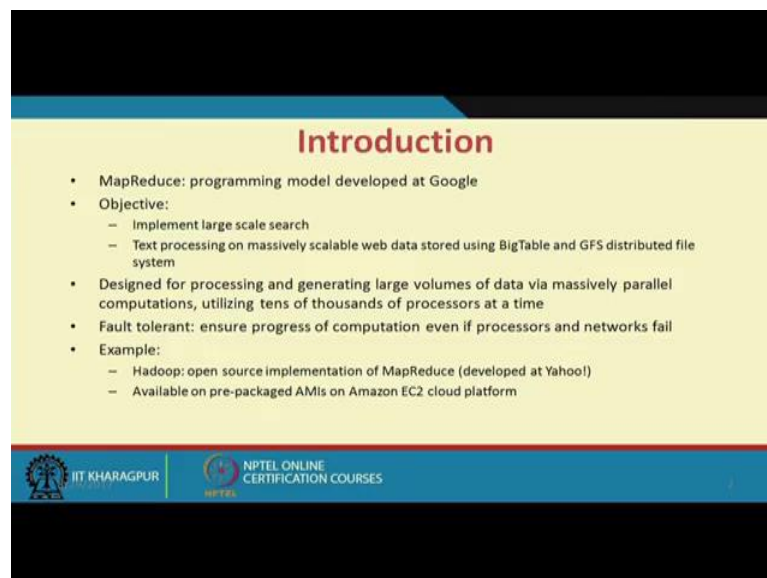
**Cloud Computing**  
**Prof. Soumya Kanti Ghosh**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 24**  
**MapReduce – Tutorial**

Hi. Today we will discuss some tutorial on MapReduce. Already we have discussed on MapReduce. So, today we will try to solve one or two problems or rather try to see how one or two; how we can decompose a problem into a MapReduce problem and how to work on it.



So, if you remember that a MapReduce paradigm is used for processing huge volume of data where paralysation is possible. And primarily rather developed by Google and later on used in various fields. So, what we will do; we will initially couple of slides we will have a quick recap before we take up one or two problems related to this MapReduce framework.

(Refer Slide Time: 01:15)



**Introduction**

- MapReduce: programming model developed at Google
- Objective:
  - Implement large scale search
  - Text processing on massively scalable web data stored using BigTable and GFS distributed file system
- Designed for processing and generating large volumes of data via massively parallel computations, utilizing tens of thousands of processors at a time
- Fault tolerant: ensure progress of computation even if processors and networks fail
- Example:
  - Hadoop: open source implementation of MapReduce (developed at Yahoo!)
  - Available on pre-packaged AMIs on Amazon EC2 cloud platform

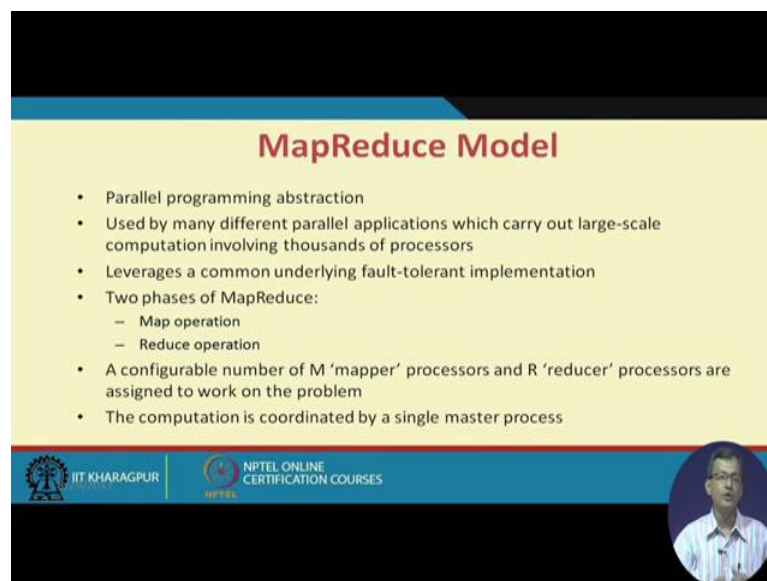
 IIT KHARAGPUR       NPTEL ONLINE CERTIFICATION COURSES

So, as we discussed already; so, it is a programming model developed at Google. Implement large scale search primarily; the basic objective was to implement large scale search. Text processing on massively scalable web data using data stored using Big Table and GFS distributed file system; Google file system and big data; Big Table.

So, this was a objective of Google which it started with; so, design for processing and generating large volume of data by massively parallel computation; utilizing tens and thousands of processor at a time. So, there are a huge number of processors to the tune of tens and thousands. So, whether if there is an inherent parallelism inside the things whether I can exploit it into the thing, so it is designed to be fault tolerant; that means, ensure progress of the computation; even if the processor or network fails. So, it should be fault tolerant to the extent that even there is a failure in the processors or the network; it to still the working should go on.

So, that was the basic assumption or basic; let us say precondition of doing this or that was basically these; what they was taken up. So, there are several things like Hadoop, open source implementation of MapReduce; incidentally it was developed by Yahoo; available on pre packaged AMIs on Amazon EC2 and so and so forth.

(Refer Slide Time: 02:45)



**MapReduce Model**

- Parallel programming abstraction
- Used by many different parallel applications which carry out large-scale computation involving thousands of processors
- Leverages a common underlying fault-tolerant implementation
- Two phases of MapReduce:
  - Map operation
  - Reduce operation
- A configurable number of M 'mapper' processors and R 'reducer' processors are assigned to work on the problem
- The computation is coordinated by a single master process

Logos: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, NPTEL, and a circular inset image of a man speaking.

So, if we look at apart from its history; so, it is a parallel programming abstraction used by many different parallel applications; which carry out large scale computation involving thousands of processors. It is again as we have doing the underlining fault tolerant implementation; both on the data side and on processor and network side. So, everything is fault tolerant; divided into two phases; one is a map phase. So, mapping the; so, the given a problem I divide into two phases. So, it is mapped into a intermediate

result and reduced to a; again, the reduce function or reduce phase; reduce those intermediate result to the actual results.

So, in doing so; what we have seen in our earlier lecture or earlier discussion on MapReduce that we can have parallel efficiency in; we can achieve substantial parallel efficiency, when we are dealing with a large volume of data and there is inherent parallelism into the process. So, what we look at there are M number of mapper processor and R number of reducer processors; which are assigned work based on the problem.

So, there is a master controller, M mapper processor and say R number of reducer cell processors which work on that. And in some cases this mapper and reducer; processor can share at the same physical infrastructure. So; that means, sometimes we acting as a mapper and at a later stage acting as a reducer type of things. So, it is a our capability of the developer or it is that how you devise these mapping and map and reduce functions. Implementation also based on that whatever that language; the developer working on, maybe there are lot of; means people are working on python, it can be on c plus plus and other type of coding things.


So, that that coding part is based on that what sort of problem and what is the environment you are working on. But primarily a philosophy there are M number of mapper and a set of reducer; you have intermediate results into the thing.

(Refer Slide Time: 05:09)

**MapReduce Model** Contd...

- Map phase:
  - Each mapper reads approximately  $1/M$  of the input from the global file system, using locations given by the master
  - Map operation consists of transforming one set of key-value pairs to another:  
$$\text{Map: } (k_1, v_1) \rightarrow [(k_2, v_2)].$$
  - Each mapper writes computation results in one file per reducer
  - Files are sorted by a key and stored to the local file system
  - The master keeps track of the location of these files

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, as we discussed earlier that each map is each mapper reads  $\frac{1}{M}$  th of the input from the global file system, using locations given by the master. So, master controller; the controller of the master node says that these are the chunks you need to read.

So, map function consists of a transformation from one key value pair to another key value pair. Like here I have a  $k_1, v_1$  mapped to  $k_2, v_2$ ; each mapper writes computation results of one file per reducer. So, it is prepared typically if there are  $R$  reducers. So, it prepares the result for one file per reducer, if there is one reducer; so one file that it creates a file for the user.

Files are stored in a key and sorted by a key and stored in a local file system. So, that is a local file systems where the output of the mapper are stored. The master keeps track of the location of this file, the master has the tracking of the things. On the reducer phase or the reduce phase; the master informs a reducer where the partial computation because the mapper has done a partial computation of the whole process; have been stored on the local file system for respective mappers.

So, if there are  $M$  mappers for the respective mappers where the files are stored for that particular reducer. Reducer make remote procedure call; request the mapper to face the files. Each reducer groups the results of the maps tape using the same key and performs a function; some function  $f$  and list of values corresponding to this value. That means, if I have as you as  $k_2, v_2$  then I map it to some  $k_2$  and function of that  $v_2$ .

So, if the function may be as simple as averaging; so, maybe frequency or the count or some complex functions of doing some more operations. So, results are written back to the Google file system, so the Google file system takes care of them.

(Refer Slide Time: 07:18)

### MapReduce: Example

The diagram illustrates a MapReduce job with 3 mappers and 2 reducers. Each mapper processes a chunk of data (D1-D8) and outputs key-value pairs. These are then grouped by key and processed by reducers to produce final word counts.

- 3 mappers; 2 reducers
- Map function:
 
$$G(k, \{w_1, \dots, w_n\}) \rightarrow [(w_1, c_1)]$$
- Reduce function:
 
$$G(w_i, \{c_i\}) \rightarrow (w_i, \sum_i c_i)$$

M = 3 mappers      R = 2 reducers

IIT KHARAGPUR      NPTEL ONLINE CERTIFICATION COURSES

So, MapReduce example; so, there are 3 mappers; 2 reducers, map function is in this case as we; if you remember or if you look at our previous lecture and discussion. So, there is a huge volume of word and what we want to do a word count. So, the every mapper has a chunk of the data things like this mapper has D 1, D 2, D 3 and this is D 4, D 7, D 8 etcetera.

So, every mapper does a partial count of the word like and for w 1, w 2, w 3 and so and so forth. And there are two reducers, so it creates file for both the reducer and so the reducer one is responsible for w1 and w ; whether the users two is for w3 and w4. And we do a word count on the thing, so there is a mapping function where this is done and there is a reducing function, where it is basically the function is for summation of this count for every word; w1. So, that is dividing this is what count problem into a MapReduce problem and last talk or last lecture we have shown that this can give parallel efficiency in the system.

(Refer Slide Time: 08:43)

**Problem-1**

In a MapReduce framework consider the HDFS block size is 64 MB. We have 3 files of size 64K, 65Mb and 127Mb. How many blocks will be created by Hadoop framework?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we now look at a couple of problems; so, this is not exactly MapReduce problem, just to go for Hadoop file system or GFS; Google file system. So, if the block size is 64 MB if you remember these file systems are larger chunk block size than never natural file system. And another thing was that there is a three replica of every instance of the data. So, there is a three replica where which allows you to have a fault tolerant mode; so based on that there are read write operations done.

(Refer Slide Time: 09:30)

HDFS Block size = 64 MB

3 files : 64Kb, 65 Mb, 127Mb

64Kb => 1		<u>Replicas (3)</u>
65 Mb => 2		
127 Mb => 2		
<u>5</u>		<u>5 x 3 = 15 blocks</u>

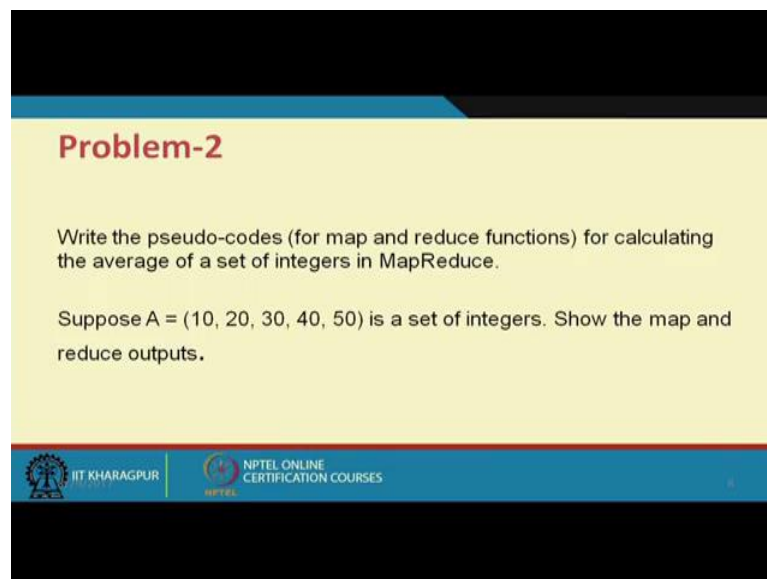
IIT KHARAGPUR

So, in this particular thing; if we say if there are; if the HDFS block size is 64 MB, then we want to find out; if there are three files of 64 K, 65 MB, Kb; MB and 127 MB.

So, how many blocks will be created by the HDFS framework. So, if for the 64 kb; how many block one will be created and 65 MB, we have 2; because upto 64 MB; 1 and 127 MB also 2. So, total 5, but in reality as there are replicas like you have replicas; so, there is typically 3 replica, so effective block size will be 5 into 3 equal to; these blocks.

So, very straightforward; nothing, no complexity in it, so if I have different type of thing. So, we can calculate this straightforward; so, again nothing to do with; immediately nothing to do with MapReduce, but nevertheless; the data is stored in either HDFS or if it is a open source or in a GFS, if it is Google file system and it need to be this data size or the storage need to be budgeted, when you are working with large data set that how much storage you require, how much storage you require to work on this type of data sets.

(Refer Slide Time: 11:29)



**Problem-2**

Write the pseudo-codes (for map and reduce functions) for calculating the average of a set of integers in MapReduce.

Suppose  $A = (10, 20, 30, 40, 50)$  is a set of integers. Show the map and reduce outputs.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us see one problem on very again very straightforward problem on MapReduce framework. So, whether we can have this MapReduce framework; though again we may not very much appreciate directly because of this simplicity of the problem. But to understand the MapReduce framework, it may be good. So you want to write the pseudo codes or codes in any language that; where there are; what we want to do? Calculate the average of a set of integers in MapReduce. So, a set of integers being pumped into the

system, it may be a direct input from the keyboard or something and so we want to find out the average of the set of integer. In other sense, in this typical case I have set of integer A as; 10, 20, 30, 40, 50.

(Refer Slide Time: 12:22)

The diagram shows the following components:

- Input Set:**  $A = (10, 20, 30, 40, 50)$
- Mapper Nodes:** 3 nodes (M1, M2, M3)
- Mapper Data:**

Mapper	Key	Value
M1	10, 20	15, 2
M2	30, 40	35, 2
M3	50	50, 1
- Intermediate Key-Value Pairs:**  $\langle 15, 2 \rangle$ ,  $\langle 35, 2 \rangle$ ,  $\langle 50, 1 \rangle$
- Reducer:** 1 reducer
  - Calculation:  $15 \times 2 = 30$ ,  $35 \times 2 = 70$ ,  $50 \times 1 = 50$
  - Sum:  $\sum \text{sum} = 150$
  - Count:  $\sum \text{count} = 5$
  - Average:  $\frac{150}{5} = 30$

So, set of integers are there so I want to make a average. So, in other sense we want to basically sum it up and divide by the cardinality. So, totally divided by 5. So in this case what we do? The master node say we consider there are three mapper. So, there are mapper nodes we considered as 3 numbers and a reducer say 1 number.

So, what do in a master node what it does for this mapper; it divides into say M1, M2, M3; 3 mapper node. So, a portion of this data say it gives 10, 20 to the first one; 30, 40 and 50. So, each mapper does a partial counting of the things; it does a averaging of these two things. So, it is something which comes up as; I can say average and count. So, first one is 15, 2; then this is 35 cardinalities 2 and this is 50; 1.

In other sense; in the temporary local file system is store 15; 2, 35; 2. What basically the output of the mapper; it does by the combiner. So, this is the map functions wants to achieve.

On the reduce; reduce is primarily is there is a one reducer it takes all the things and it does a averaging of the things or more this averaging of the whole thing. So, in other

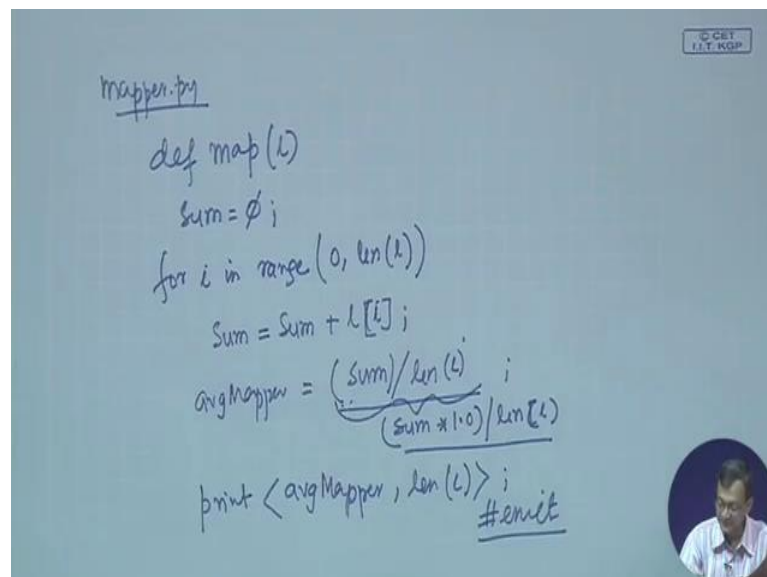


sense what it does say 15 star 2; here 35 star 2 and 50 star 1. In other sense is sigma sum is 150; sigma count is 5.

So, it says 150 by 5 are 30; what it is basically 15 star 2 plus 35 star 2; so, that the exactly does. So, the problem is pretty straightforward or simple you may not find that; what is the big deal here? If the number of things is pretty high coming as a stream, and then I can basically do a parallel things; these are parallely processed and this is reduced by the one particular reducer.

So, if we write the code for it; so, you can use any language to do that. Here we are using say python or python type language, it is the language is does not matter; the representation you can do use any pseudo code and type of things.

(Refer Slide Time: 16:21)



```
mapper.py
def map(l)
    sum = 0
    for i in range(0, len(l))
        sum = sum + l[i]
    avgMapper = (sum/len(l)) / ((sum * 1.0)/len(l))
    print <avgMapper, len(l)>
    #emit
```

So, you have that mapper function or say mapper dot py; so, it is something python type. So, we are not much giving importance to the syntax; rather we are more giving importance to the concept.

So, def map; so l is the list; so, what we do? We initialize a sum equal to 0 for i in range 0 to length of l, sum equal to sum plus. So, every mapper does this; every mapper what it does; it takes that chunk of data which is being allocated to it for by the master node and then in our cases how many data is there?

The mapper 1 has 2 data; mapper 2 has 2 data mapper 3; M 3 has 1 data; so, it is 2; 2 each. So, for every mapper we done this; so, we that average sum by length of l in this case 2; it should be like this sum by length of 2. Or in other more strictly speaking, we should; to have a floating point difference, floating point divide because this is a; otherwise there may be integer divisions.


So, we can ideally give some star 1.0 divided by length of l. So, length of l; in this case 2 and then we output this; let us use print function, output this to the local file system. So, there can be this command wise; it may be different if you are using different programming paradigm lengths; so, the mapper basically emit these data.

So, it is stored in the local file system; so, what we are doing, for every mapper we are reading a list of data which is being assigned by its master node, making that some initializing the sum to value, then we add what we are doing for i; for in the loop. So, calculating the sum making a average out of it rather this is nothing, but to make it float division. And then it is emitting or dumping that value into the local file systems; which the reducer will read it.

So, this is the mapper portion of the thing; so, if we look at the reducer portion what we have def reduce. So, what it reads; so, whatever the mapper has dumped in the things. So, if you look at it is giving these average value and the lengths of the thing.

(Refer Slide Time: 20:14)

```
reducer.py
def reduce (avg, l)
    sum = count = 0;
    for i in range (0, len(l))
        sum = sum + avg[l[i]] * l[i]
        count = count + l[i]
    average =  $\frac{sum * 1.0}{count}$ 
    print <average>;
```



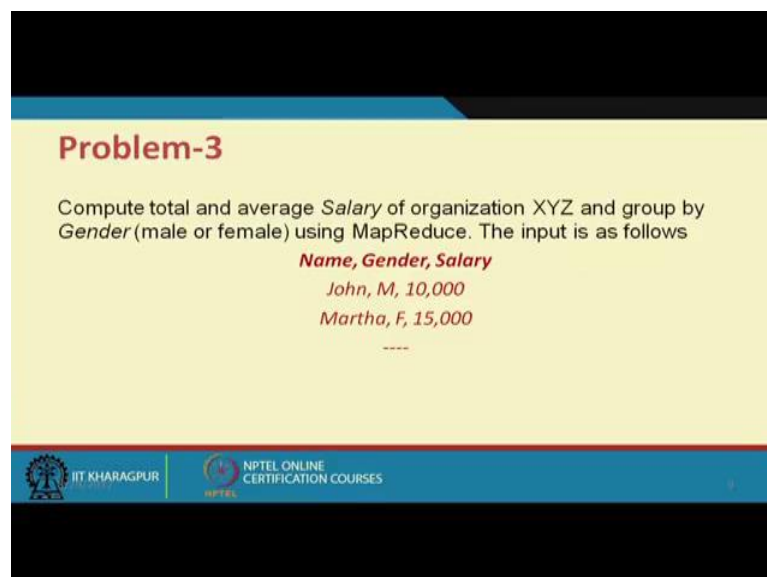
Here also we read that particular thing sum equal to; so, here for i in range of 0; length of l. So, it is what it is doing if we look at our previous thing; so, what it is doing, it is basically trying to calculate this sort of values. So, count also equal to count plus length of i; so, finally what we get average. Again sum to make it float; yes multiply this or you can basically typecast also count and then print average it.

So, what it is reducer is doing taking this local file system as there are only one reducer. So, it takes all the values and for all that data; it goes on summing up that output from the each mapper, in this case there are 3. So, it is it is coming to be 15 into 2; plus 35 into 2 plus 50 into 1; divided by count, which is here 2 plus 2 plus 1 is 5. So, and it calculates the average value and then it again writes the average value to the Google file system or Hadoop file system based on the whatever the requirement is this.

So, here this is that again though this is may be a straightforward simple thing, but we see that I can divide a problem; as there are inherent parallelism like there are; I could have like in order to do a averaging I have taken a chunk of data and that we try to solve it using in MapReduce framework.

So, if there is a huge volume of data then the mapper; that the master node divides accordingly and do the partial computation and the reducer read it from and do the final computation. So, this is again a simple example of a MapReduce framework; so, next we see another problem.

(Refer Slide Time: 23:41)



**Problem-3**

Compute total and average *Salary* of organization XYZ and group by *Gender* (male or female) using MapReduce. The input is as follows

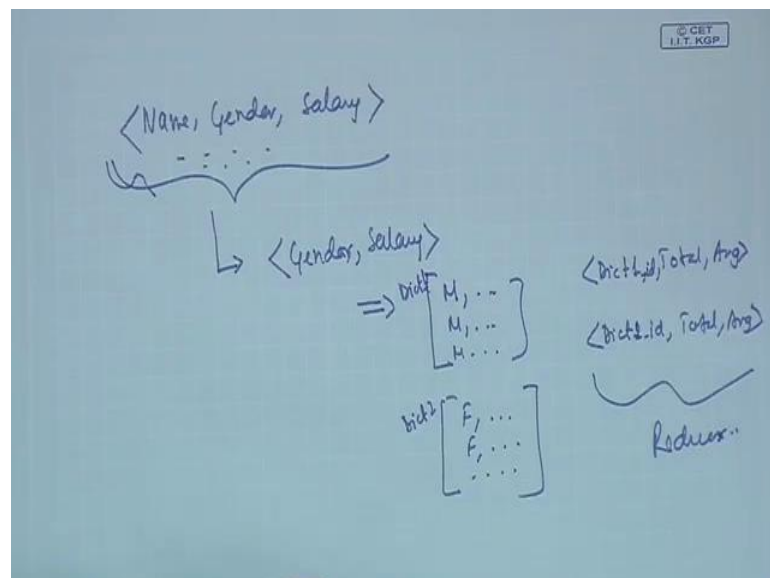
<i>Name</i>	<i>Gender</i>	<i>Salary</i>
John	M	10,000
Martha	F	15,000
----		

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what it says I want to compute the total and average salary of organization XYZ; some organization grouped by the gender using MapReduce. So, input is name, gender and the salary of the thing; in this case say name is John, gender is M or male and salary is something 10000 unit or maybe 10000 dollar or something. And the next one is Martha, gender is F and salary is something 15000.

So, what we want to do? We want to find out that male wise, like gender wise; in this case male and female that what is the your total and average salary; anyway that total divided by the cardinality will be the average salary of the things; so, the output will be like in that form. So, what we try to look at whether we can employ MapReduce problem to look into this particular problem. So, let us look at it.

(Refer Slide Time: 24:53)



So, what we are having? We are having this tuple like this; name gender and salary. So, this is the tuple; so, what we want to do in the map phase? So, want to if the input data whatever the input data say it is there; there are different set of the input data set. So, what we want to do? We want to extract only because we are not bothered about the gender M; bothered about the name of the person or that is not required in the; so, in the salary.

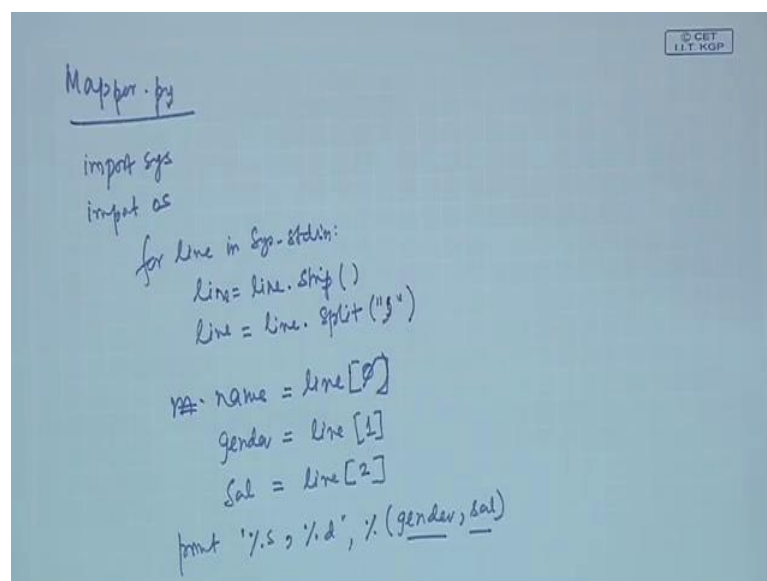
So, we want to calculate from there these two thing say M the salary; so, respective and so, they want to do type of this type of things or I can have a key value pair. Key is this; male or female and value is the salary of the things or we can say that we have two sort

of a dictionary structure which is having a key value pair and then having say; I say this is Dict 1, Dict 2 and these two type of key value pair and for every 1, I can have Dict 1 then maybe total an average for other one also Dict 2; id maybe.

So, so id in this case is this particular male or female and then having total or average salary. So, compute total average for the two separate say we consider there is a dictionary structure and the reducer basically does this thing. So, want to see that how to realize this, so what let us look at the problem.

So, I have a set of name, gender and salary; want to extract that gender and the salary from every topple. So, if I have multiple mapper; so, I extract those things and dump as a particular two type of dictionary type. One is that two type of thing; one is that what we have this with M and f and the at the reducer part, we calculate the total and average salary of the things or any of the thing.

(Refer Slide Time: 28:01)



```
Mapper.py
import sys
import os
for line in sys.stdin:
    line = line.strip()
    line = line.split("\t")
    name = line[0]
    gender = line[1]
    sal = line[2]
    print '%s %d', %(gender, sal)
```

So, again we look at as a mapper dot py or mapper dot some python type code. So, again I am just want to again repeat it; that you can do with any coding language which is suitable for this and whatever we are doing may not be; there may be some syntactical problem with the actual python thing, but it does not matter the conceptually you want to show that things works and then actual syntactical syntax need to be followed, if you are really want to implement this.

So, this is the thing for a line in sys dot std in; what we are doing. So, what we are considering that it is separated by a comma. So, we are split is; sorry it should be comma. So, I now separate name equal to line 0, salary represents Sal is; line 2 while generating the; or emitting the mapper phase, the data into the local file system. So, we keep print comma percentage d.

Then, so what we do? Gender and salary; in other sense this syntax you need to check up. In other sense, what we do we dump basically gender in the salary into the thing or M and the salary portion; like as you see we want to generate this M and the salary portion and another thing is that either M or f and the salary portion.

(Refer Slide Time: 31:22)

```

Reducer.py
import sys
dictorg = {}

for line in sys.stdin:
    line = line.strip()
    gender = line[0]
    sal = line[1]

    if gender in dictorg:
        dictorg[gender].append(int(sal))
    else:
        dictorg[gender] = []
        dictorg[gender].append(int(sal))

for gender in dictorg.keys():
    SalAvg = sum(dictorg[gender]) / len(dictorg[gender])
    TotalSal = sum(dictorg[gender])
    print '%s, %.d, %.d, %.d' % (gender, TotalSal, SalAvg, len(dictorg[gender]))
    
```

So, in the reducer phase; what we do that import; so, define that or call this dict org that is dictionary class for line in sys in what we do. So, what it is reading? It is reading basically that the gender or that key value pair with the gender and the value of the things or in other sense the gender and the salary values. So, we do not have that name into the things because this particular query does not require the name of the thing; line of 1.

So, if it is already existing; that means, once you have read then dict org; so, what is our objective? So basically sum up the salaries by adding go on adding on the salary values for the same gender type. So, already if there; that means; so, what that now my existing the reducer dictionary counting, a key value pair. So, if the key is already that gender is

there or male or female, then I go on adding those things whenever I get. If it is not there if it is else; that means, this is basically the initialization thing dict org.

So, initialize with a blank thing; so first time when it is coming. So, in first time it is coming; that means, there is a blank thing. So, if it is blank then it basically initialize; then append the salary; that means, it is initialized with the salary of the dict org dot keys; salary average equal to sum of dict org; gender divided by length of dict org gender. So, it is summing up divided by things straightforward and total salary equal to only sum of dict org gender. And then we basically write back the other thing from to the Google or GFA or the HDFS file system. If we want to separate it by a comma or tab as the case may be; again maybe D if it is a integer or based on that if it is a float and all those things.

So, we have this as gender total sal and salary avg. So, that is the what we do at the final reduce surface. So, if we try to just quickly have a look, so what we are doing in the mapping function; we have three thing like name, gender and salary. Our objective is to the mapping functions; so, see this all the map are we like find out this individually this whether it is a; which gender M and find keep this salaries along with that g M or f and salary and the reducer will basically; so, that it exactly that gender and salary and the reducer will basically extract that intermediate result and calculate the average and the total.

So, here that that operation is there; so, this is a typical python type, I am not strictly telling python because there may be some syntactical issue. But you can implement in anything, the idea is that I divide the problem into smaller parallel things by the mapper and then in the second phase; the reducer put it to another key value pair. So, key value from the input set; to a set of key value pair, reducer takes that key value pair and put a function; in this case average or total of the things, to another set of key value pair and the finally, it goes to the HDFS pair or GFS file system ok.

So, what we tried to look at in today's thing that; this MapReduce functions say simple problems; how we can put it into map and reduce things. That this number of mappers available; allow is basically availability of the resource and the how the master nodes divide it and the number of reducers also based on the term. What type of functional

things you want to do and so they master node is there, it divides into M number of mapper and a number of reducer.

The problem is the functionality of the problem is divided in such a way so that it can be executed in two phases, and we can have a parallel implementation of this sort of paradigm.

Thank you.