SOLUTION:

It has been given that there are sqrt(n) machines and each machine has 10 * sqrt(n) of memory space. It has also been given that there are n items with a range of 1 to n, which are distributed among sqrt(n) different machines.

The output to be calculated should be of the form (i, summation[v_j where $j \leq i$]).

Now, consider that we are defining two different functions called Map() and Reduce(). Given that there are n different items and their associated values, they can be divided up into sqrt(n) chunks of data items arbitrarily as mentioned in the problem. In the map function each item can be identified using its "I" value and its corresponding "v" value can be determined.

But since we have to compute the summation of "v" values of an item i, we should also consider values of items that preceded "I" in a number line. For example, if i=5 then v[5] = v[5]+ v[4]+ v[3]+ v[2]+ v[1].

Let us define a map function. The pseudo code is as follows:

```
func Map(i,V)

{

    repeat
        M[i-1] = V[i-1]    // Since array indexes start from 0, for every i value the associated value lies
                                                                       in an array with an index of i-1
                           // here V[] is the array consisting of all the values from i=1 to n
    i = i-1
    until(i=0)
   return (i,M)    // M is an intermediate array that will be given as an input to the reducer function
}
```

In the above pseudo code, V[] is the array that contains all the values from i=1 to n. Then an intermediate array is created which contains only those values till the given "i" value. For example, if i=4, then the function returns (4, {v[4],v[3],v[2],v[1]}). This returned output will later on be used for the Reduce() function.

After the map function has been executed, we get the output of as n output pairs where each "i" value has a value list associated to it. The value list comprises of v[i] and all other item values preceding i. For example, i=5 will consist of {v[5],v[4],v[3],v[2], v[1]} i.e, (i,{v[5], v[4],v[3],v[2], v[1] }. Now, we can sort the (item,value-list) obtained into sqrt(n) chunks, each chunk containing items sequentially(this step is not really necessary) and then arbitrarily assign these chunks to sqrt(n) machines, as required by the given problem.

Now, we can define and use a Reduce() function which will take an item and its associated value list from the chunk that it has been assigned and then compute the summation of all the values in the value-list. For example, (i,{v[5]+ v[5]+ v[4]+ v[3]+ v[2]+ v[1]}).

The Reduce() function is defined in pseudo code as follows:

```
func Reduce(i,M)
```

```
{
  res=0    // Res variable is assigned to 0 initially
  j=0
        repeat
                res= res+M[j]
                j=j+1
        until(j=i-1)
        print i
        print res
        return 0
  }
```

In the above Reduce () function, an item and its associated value list is taken as input parameters and the summation of all the values of the list are performed and the "i" and "res" are printed as outputs.

Thus we get the output as follows:

(1, v[1])

(2,{v[1]+v[2]})

(3,{v[1] +v[2] +v[3]}) . . . and so on.

In the above approach, the time complexity involved is O(n). Here the memory taken up will not exceed the machine capacity of 10 * sqrt(n), since we are assigning only sqrt(n) values and their lists to each machine.

Thus, at each round the Map function gives out : (item,{value-list}) and the Reduce function gives out : (item, summation of all value list values)

REFERENCES:

1 . Database Systems, the complete book (Ullman)

2. MapReduce and New Software Stack - http://infolab.stanford.edu/~ullman/mmds/ch2.pdf

3. MapReduce Tutorial – hadoop.apache.org

DISCUSSION: This assignment was discussed with the following students:

1. Rohit Nedunuri
2. Prerit Anwekar