MATRIX-VECTOR MULTIPLICATION WITH MAP/REDUCE

PROFESSOR TAHAR KECHADI

MAP/REDUCE PROGRAMMING MODEL

- Map Function
  - Input: (key, value) pair
  - Output: set of intermediate (key, value) pairs
- Reduce Function
  - Input: (key, list of values) → intermediate key and set of values for that key
  - Output: (key list of values) → key in the output is identical to the key in the input

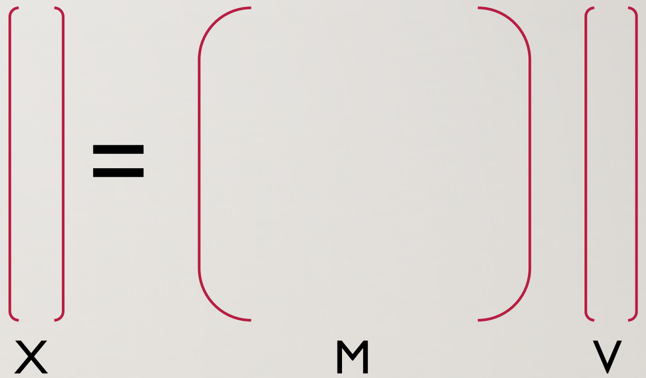
MAP/REDUCE PROGRAMMING MODEL

- mappers group together pairs with the same key
- Mappers pass to the reducers (key, value) pairs
- The mappers can be executed in parallel
- The reducers can be executed in parallel
- All the mappers need to finish before the reducers can start
- MapReduce program may consist of many rounds of different map and reduce functions

MATRIX-VECTOR MULTIPLICATION

- Consider a $M(n \times n)$ and a vector $V(n)$
- The objective is to calculate a vector $X(n) \;=\; M \times V$

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$



- Each mapper can load the vector V
- Map Function: maps $\left((i,j), m_{ij}\right)$ to $(i, m_{ij} v_j)$
- Reduce function: receives $(i, [m_{i1} v_1, \dots, m_{in} v_n])$:sums all values of the list of a key i, to produce (i, x_i)

MAP/REDUCE ALGORITHM FOR THE EXAMPLE

- The Map Function
- For each element m_{ij} do
- produce (key, value) pairs; $((i, j), m_{ij})$ as $(i, m_{ij}v_j)$ for $j = 1,2,3, \dots, n$
- Return set of (key, value) pairs; $(i, m_{ij}v_j)$ for all possible values of j,

MAP/REDUCE ALGORITHM FOR THE EXAMPLE

- The Reduce Function
- For each Key (i) do
-         sum up $m_{ij} \times v_j$ for $j = 1,2,3, \dots, n$
- Return $(i, \sum_{j=1} m_{ij} \times v_j)$

```
Public class MatrixVector_Multiply {
  public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println(“Usage:MatrixVector_Multiply <in_dir> <out_dir>”);
            System.exit(2);
        }
        Configuration conf  = new Configuration();// M is n x n matrix,V is an n-vector.
        conf.set(“n”,“1000”);
        Job job = new Job(conf,MatrixVector_Multiply”);
        job.setJarByClass(MatrixVector_multiply.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class)
        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.addOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
    }
}
```