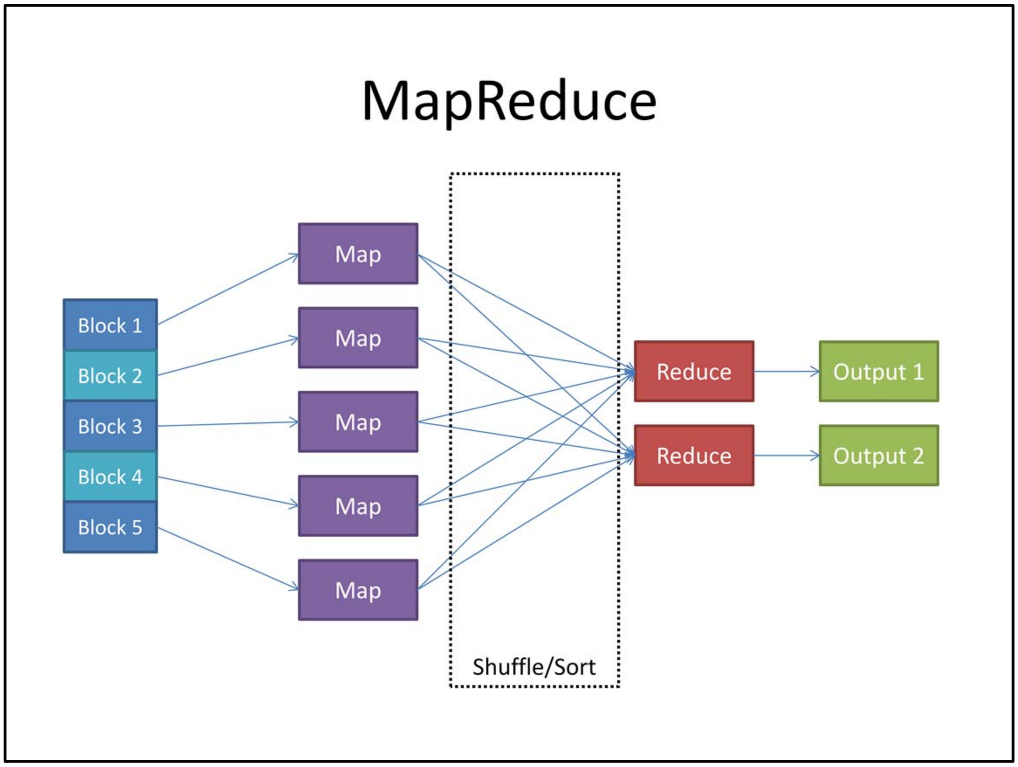
Sort in MapReduce

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

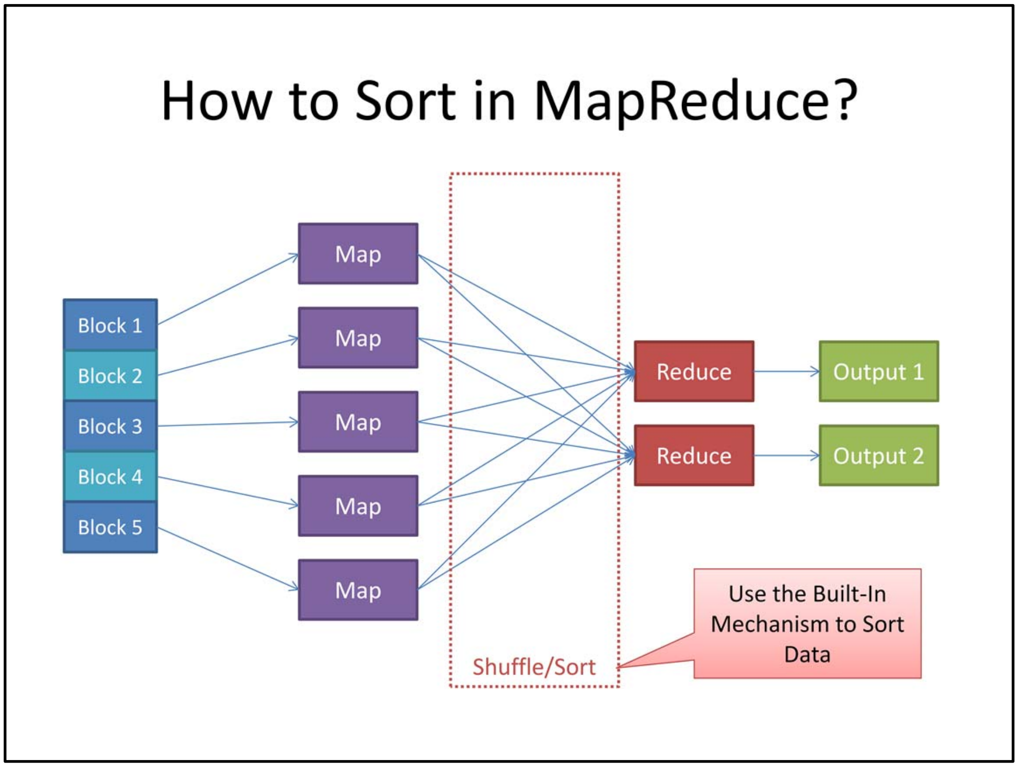
In this video, we will look at a typical implementation of a “Sort” in MapReduce.



MapReduce

Block 1
Block 2
Block 3
Block 4
Block 5
Map
Map
Map
Map
Map
Shuffle/Sort
Reduce
Reduce
Output 1
Output 2

As you may recall, MapReduce program expect input to be split into chunks, this is typically done by HDFS in the case of Hadoop. Each HDFS block is sent to a Map task which transforms the input into an intermediate output. All of the intermediate outputs from all the mappers are sorted and sent to the reducer, which may perform another computation and write output.



How to Sort in MapReduce?

Block 1
Block 2
Block 3
Block 4
Block 5

Map
Map
Map
Map
Map

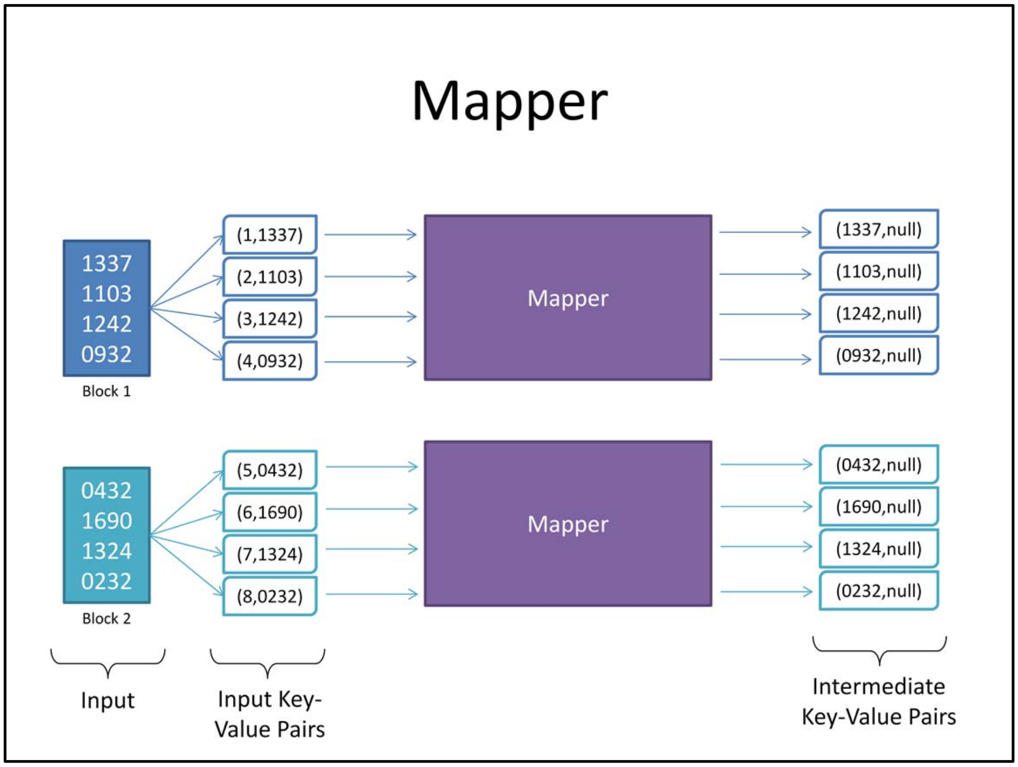
Shuffle/Sort

Reduce
Reduce

Output 1
Output 2

Use the Built-In Mechanism to Sort Data

In order to implement Sort in MapReduce, all we have to do is to use the built-in mechanism to sort the data, using the Shuffle/Sort phase. Let’s look at a more concrete example of how this works.

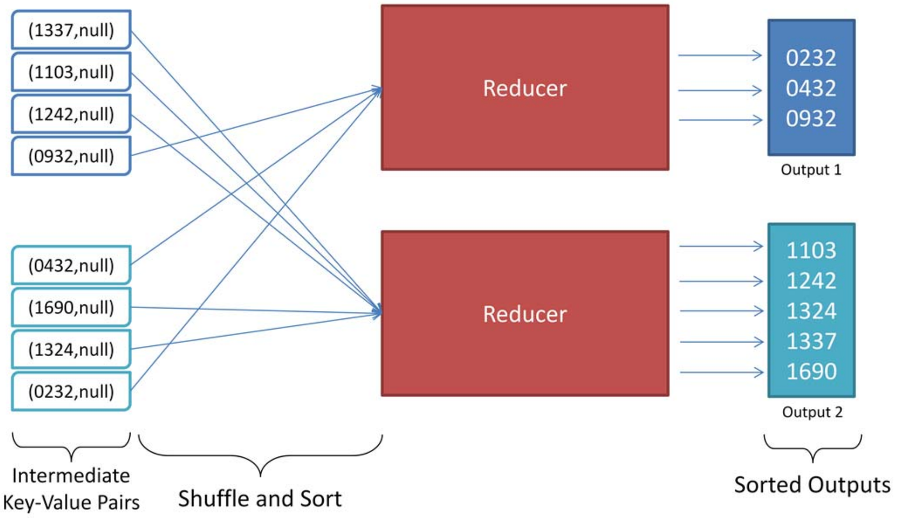


Mapper

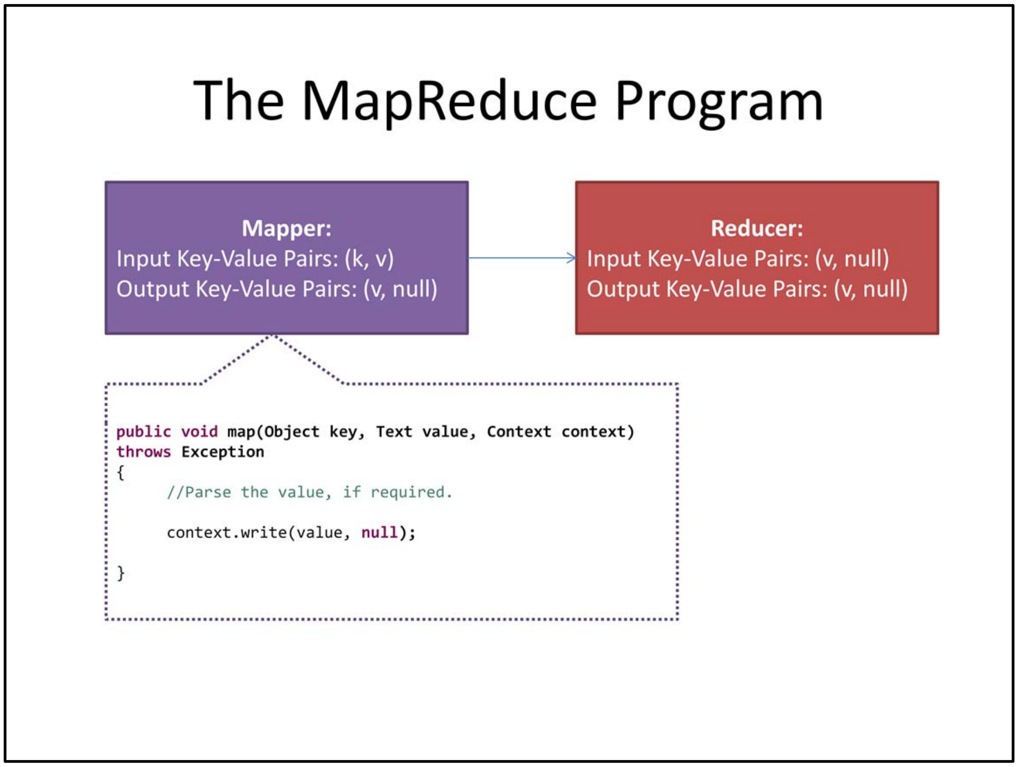
Let us assume two input blocks, 1 and 2 contain the data to be sorted. For this illustration, we have a few 4-digit numbers to be sorted. Each of these numbers are transformed into input key-value pairs though an InputFormat. In the illustration assuming the standard TextInputFormat we have the byte offset as a key, followed by the value which is the actual value of the line.

The Mapper in this case simply transforms the input key value pairs by discarding the input key, using the input value as the intermediate key and passing no intermediate value.

Reducer



The intermediate key-value pairs are shuffled by key to the corresponding reducers. In this example, we assume that the partitioner function is aware of the data. Through the shuffle and sort, the reducer accepts sorted intermediate key value pairs and simply writes them straight to disk.



The MapReduce Program

Mapper:
Input Key-Value Pairs: (k, v)
Output Key-Value Pairs: (v, null)

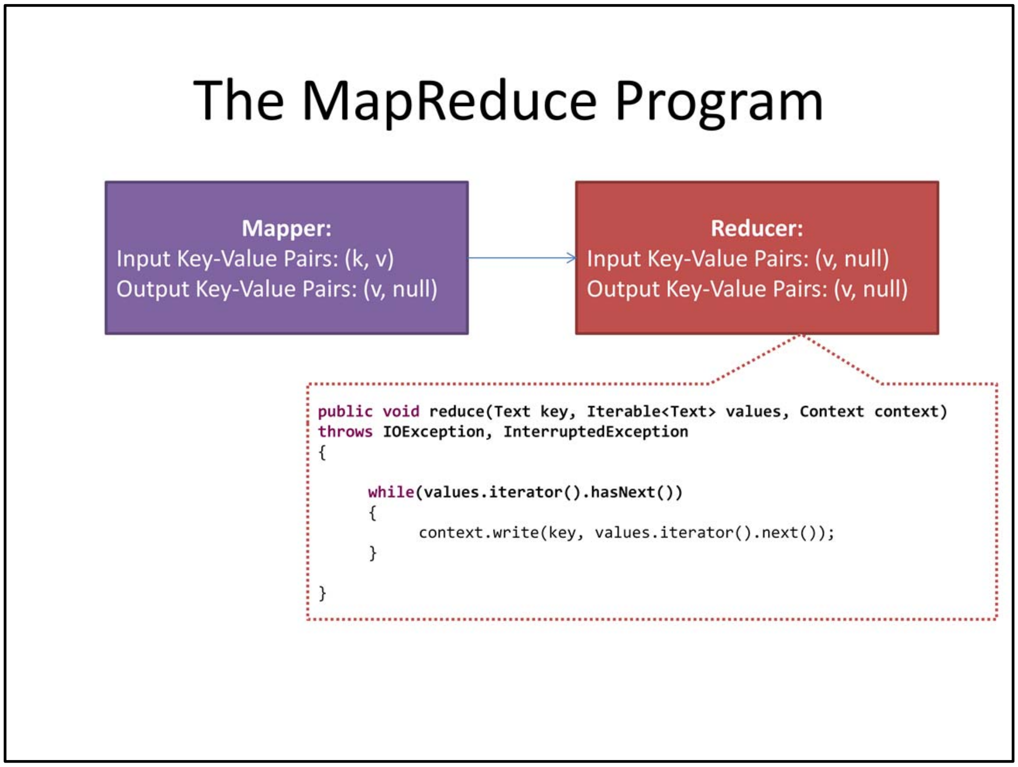
Reducer:
Input Key-Value Pairs: (v, null)
Output Key-Value Pairs: (v, null)

```
public void map(Object key, Text value, Context context)
throws Exception
{
     //Parse the value, if required.

     context.write(value, null);

}
```

On screen are the mapper implementations, and the reducer implementations of the program.



The MapReduce Program

Mapper:
Input Key-Value Pairs: (k, v)
Output Key-Value Pairs: (v, null)

Reducer:
Input Key-Value Pairs: (v, null)
Output Key-Value Pairs: (v, null)

```
public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException
{

     while(values.iterator().hasNext())
     {
          context.write(key, values.iterator().next());
     }

}
```

On screen are the mapper implementations, and the reducer implementations of the program.