MapReduce

Tushar B. Kute,
http://tusharkute.com

What is MapReduce?

- MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.
- MapReduce is a processing technique and a program model for distributed computing based on java.
- The MapReduce algorithm contains two important tasks, namely Map and Reduce.

Map and Reduce

- Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).
- Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

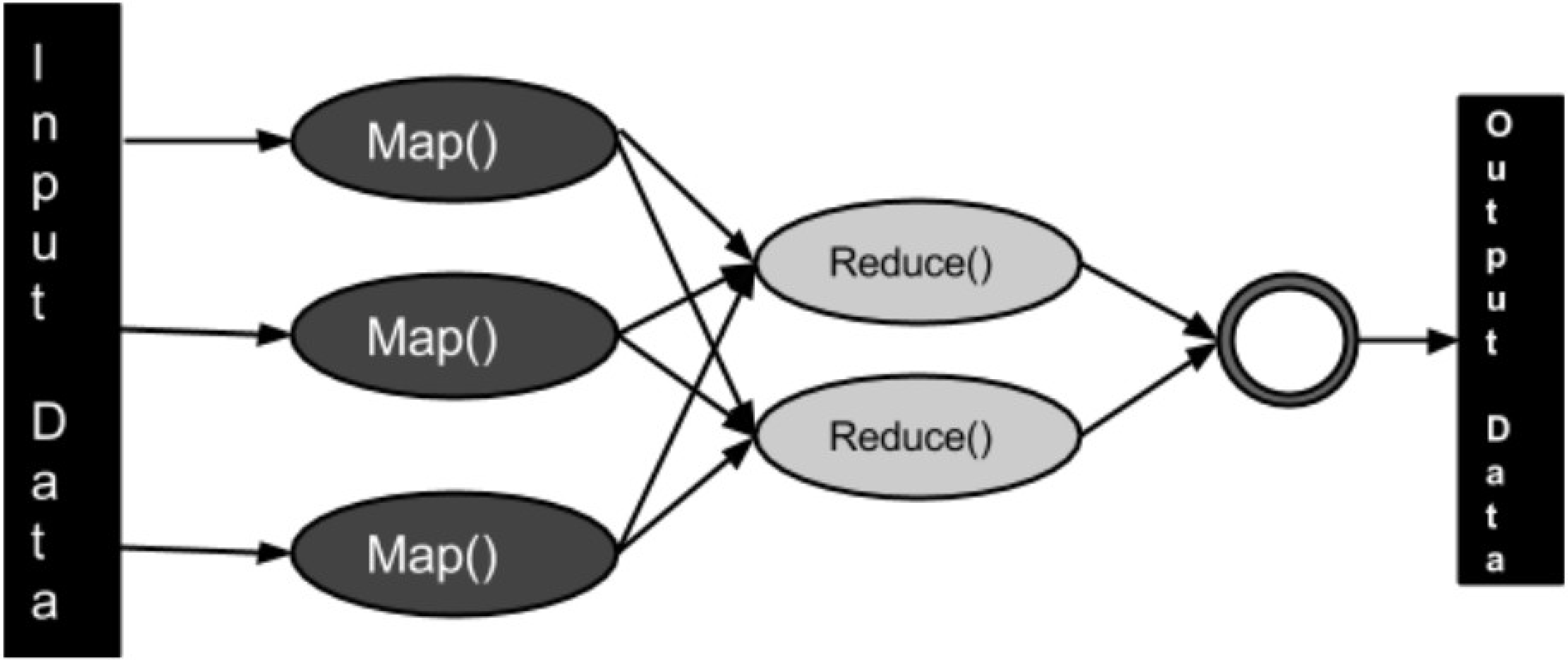
Map and Reduce

- The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.
- Under the MapReduce model, the data processing primitives are called mappers and reducers.
- Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.
- This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
- **Map stage**: The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
- **Reduce stage**: This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

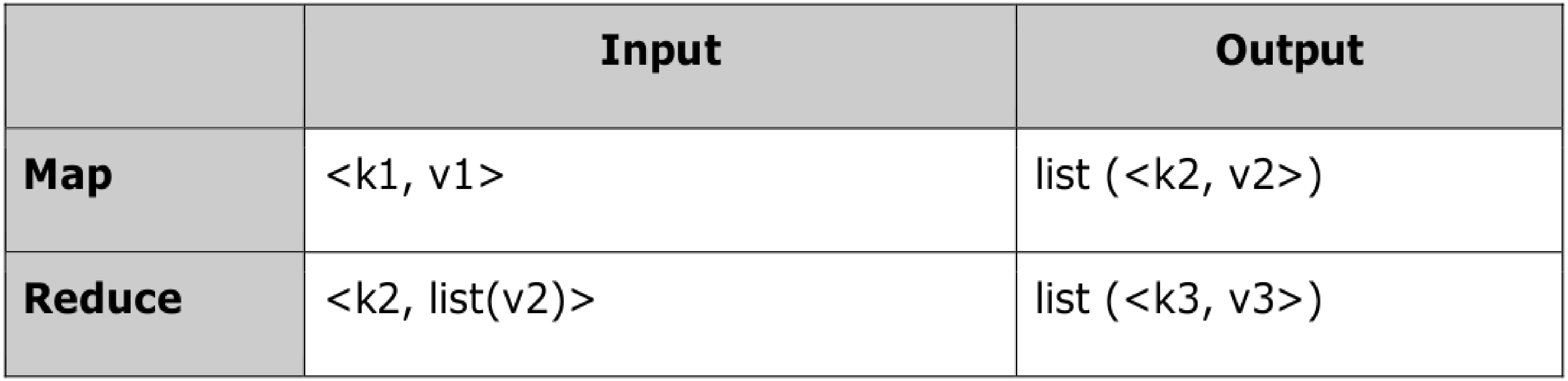
The MapReduce



Inserting Data into HDFS

- The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.
- The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework.
- Input and Output types of a MapReduce job: (Input) <k1,v1> -> map -> <k2, v2>-> reduce -> <k3, v3> (Output).

Shutting Down the HDFS



	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

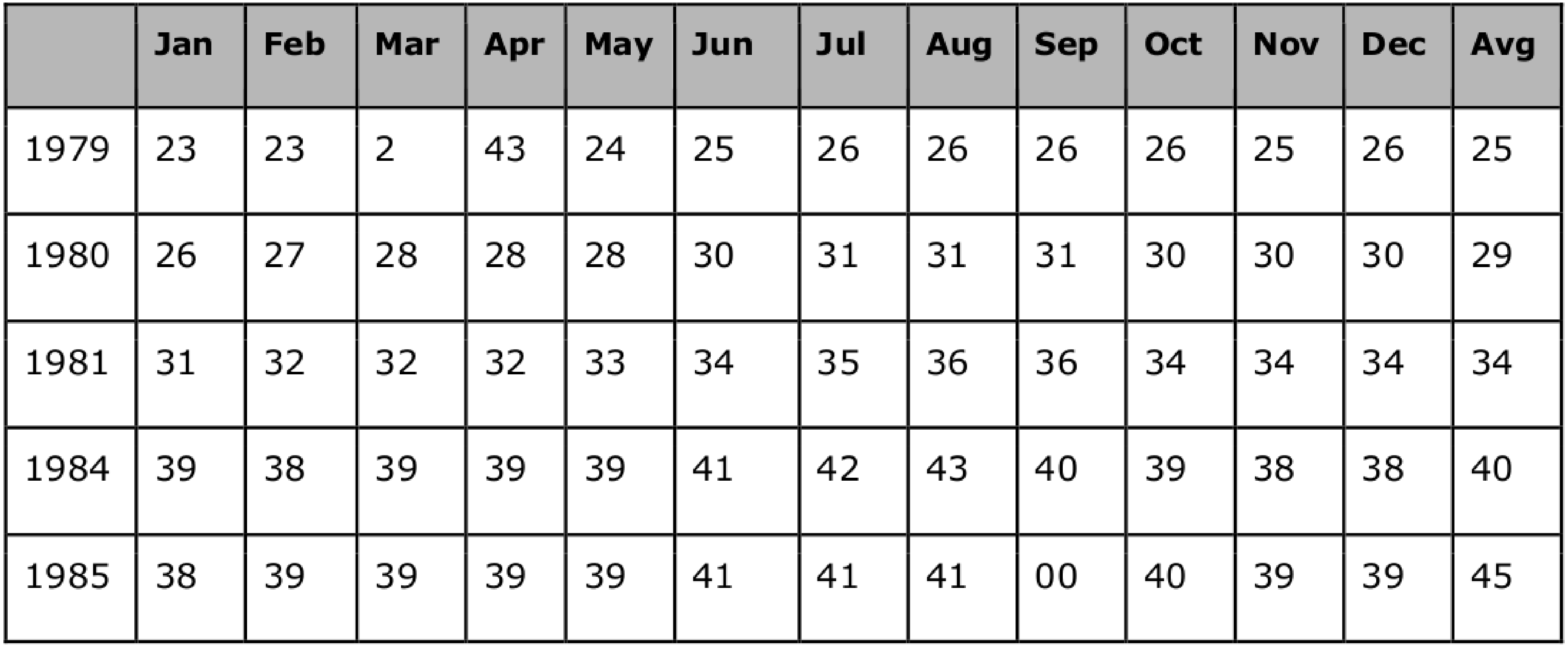
Terminology

- PayLoad - Applications implement the Map and the Reduce functions, and form the core of the job.
- Mapper - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- NamedNode - Node that manages the Hadoop Distributed File System (HDFS).
- DataNode - Node where data is presented in advance before any processing takes place.

Terminology

- MasterNode - Node where JobTracker runs and which accepts job requests from clients.
- SlaveNode - Node where Map and Reduce program runs.
- JobTracker - Schedules jobs and tracks the assign jobs to Task tracker.
- Task Tracker - Tracks the task and reports status to JobTracker.
- Job - A program is an execution of a Mapper and Reducer across a dataset.
- Task - An execution of a Mapper or a Reducer on a slice of data.
- Task Attempt - A particular instance of an attempt to execute a task on a SlaveNode.

Example:



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

Example:

- ProcessUnits.java

Compilation and Execution

- Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).
- Follow the steps given below to compile and execute the above program.
- Step 1
  - The following command is to create a directory to store the compiled java classes.
  -
    ```
    $ mkdir units
    ```

Compilation and Execution

- **Step 2**
- Download Hadoop-core-1.2.1.jar, which is used to compile and execute the MapReduce program. Visit the following link

  http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1

  To download the jar. Let us assume the downloaded folder is /home/hadoop/.
- **Step 3**
- The following commands are used for compiling the ProcessUnits.java program and creating a jar for the program.
- `$ javac -classpath hadoop-core-1.2.1.jar units/ProcessUnits.java`
- `$ jar -cvf units.jar -C units/ .`

Compilation and Execution

- Step 4
  - The following command is used to create an input directory in HDFS.

  -
    ```
    $HADOOP_HOME/bin/hadoop  fs  -mkdir  input_dir
    ```
- Step 5
  - The following command is used to copy the input file named sample.txt in the input directory of HDFS.

  -
    ```
    $HADOOP_HOME/bin/hadoop fs  -put  /home/hadoop/sample.txt
    input_dir/
    ```
- Step 6
  - The following command is used to verify the files in the input directory.

  -
    ```
    $HADOOP_HOME/bin/hadoop  fs  -ls  input_dir/
    ```

Compilation and Execution

- Step 7
  - The following command is used to run the Eleunit_max application by taking the input files from the input directory.
  - `$HADOOP_HOME/bin/hadoop jar units.jar ProcessUnits input_dir/ output_dir/`
  - Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

Compilation and Execution

- Step 8
  - The following command is used to verify the resultant files in the output folder.
  - `$HADOOP_HOME/bin/hadoop fs -ls output_dir/`
- Step 9
  - The following command is used to see the output in Part-00000 file. This file is generated by HDFS.
  - `$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000`
- Below is the output generated by the MapReduce program.

```
1981 34
1984 40
1985 45
```

Compilation and Execution

- Step 10
- The following command is used to copy the output folder from HDFS to the local file system for analyzing.
  - $HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs -get output_dir /home/hadoop

Commands

- All Hadoop commands are invoked by the $HADOOP_HOME/bin/hadoop command. Running the Hadoop script without any arguments prints the description for all commands.
- Usage:
  - hadoop [--config confdir] COMMAND

Commands

- **`namenode -format`**
  - Formats the DFS filesystem.
- **`secondarynamenode`**
  - Runs the DFS secondary namenode.
- **`namenode`**
  - Runs the DFS namenode.

Commands

- **`datanode`**
  - Runs a DFS datanode.
- **`dfsadmin`**
  - Runs a DFS admin client.
- **`mradmin`**
  - Runs a Map-Reduce admin client.
- **`fsck`**
  - Runs a DFS filesystem checking utility.
- **`fs`**
  - Runs a generic filesystem user client.
- **`balancer`**
  - Runs a cluster balancing utility.

Commands

- `jobtracker`
  - Runs the MapReduce job Tracker node.
- `tasktracker`
  - Runs a MapReduce task Tracker node.
- `job`
  - Manipulates the MapReduce jobs.
- `queue`
  - Gets information regarding JobQueues.
- `version`
  - Prints the version.
- `jar <jar>`
  - Runs a jar file.

Thank you

Web Resources
http://mitu.co.in
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com