Parallel Computing with R - MapReduce Computation

Laurence Viry

Grenoble RUG - Working sessionn°8 - Parallel Computing

26 Avril 2018

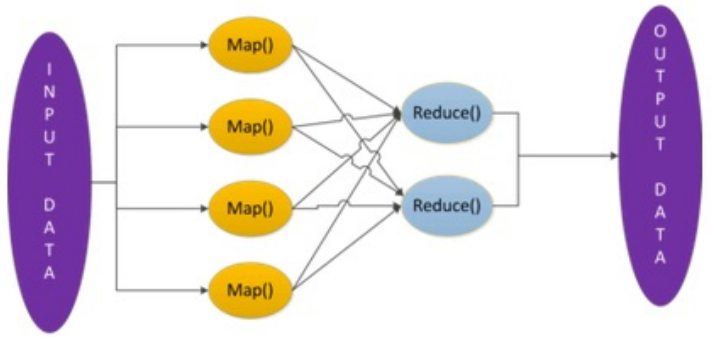
MapReduce Computation: motivations

- Into a era of **Big Data**, demand grew for a computing paradigm that:
  - is generally applicable,
  - work on **distributed data.**
- **Distributed data** means that data is **physically distributed**
  - over **many chunks,**
  - **possibly on differents disks,**
  - **may be even different geographical locations.**
- **Data stored in a distributed manner**:
  - facilitates parallel computing,
  - parallel chunks can be read simultaneously,
  - enable to work with data sets too large to fit into memory of a single machine.

Such computational capability led to the use of the MapReduce paradigm.

MapReduce paradigm

MapReduce is a form of the scatter-gather pattern with a sorting operation added in the middle. The input is in a distributed file, fed into the following process.



- **Map Phase**: various processes (**mappers**). For each line of the input file, the mapper read the line, processes it in some way and emits an output line under the form (key,value).
- **shuffle/sort phase**: all the output lines that share the same key are gathered togather.
- **Reduce phase**: various parallel process (**reducers**), all running the same code. Each reducer work on its own set of keys.

Apache Hadoop

The most popular MapReduce sofware package is **Hadoop**, written in Java, used with the langage C++, it includes a streaming features that enable to use Hadoop from any language, including R.

- Hadoop includes its own distributed file system called Hadoop Distributed File System (**HDFS**) which is replicated, thus achieving some fault tolerance.
- Input to the mappers is from HDFS file, and output from the reducers is to an HDFS file, one chunk per reducer.
- The file line format is: **key** `\t` **data** where `\t` is the Tab character.

MapReduce implements the following features:

- Automatic parallelization of Hadoop programs.
- Transparent management of the distributed mode.
- Fault tolerance.

More generally, MapReduce greatly simplifies the life of the developer Hadoop, by masking a good part of the inner workings of Hadoop.

R interfaces to MapReduce system

- There are several RHadoop packages that one has to use in order to be able to connect R and Hadoop.
  - **rhdfs**: provides commands for file manipulation in terms of reading, writing and moving files from hadoop filesystem.
  - **rmr2**: allows usage of Hadoop MapReduce facility inside the R environment.
  - **plyrmr**: provide a wide palette of predefined operations to cover basic data manipulation needs.
  - ...
- To work with R and Hadoop, the first step is to integrate R with Hadoop

  Start the Hadoop server:

  start-dfs.sh
  start-yarn.sh

  In R:

  library(rhdfs)
  hdfs.init()

  Stop the Hadoop server by using counterpart commands stop-yarn.sh and stop-dfs.sh in terminal.

Example

- **Word Count**: word count for a text file.
  - **mapper**: breaks each line into words, and emits (key,value) pairs in the form (word,1) (wordmapper.R).
  - **reducer**: all thise 1s for a given word are summed, giving a frequency count for that word (wordreducer.R).
- **Running a code**:
  - I need to place my data file in HDFS.

    hadoop fs -put ../data/rnyt rnyt
  - Running in streaming mode the mapreduce R code

    hadoop jar contrib/streaming/*.jar -input rnyt -output countrnyt -mapper wordmapper.R -reducer wordreducer.R
- **Computing centroids of big data** using mapreduce from rmr2.