COMP5349 – Cloud Computing

Week 1: Homework Tips

Uwe Roehm
School of Information Technologies

Week 1 Programming Homework

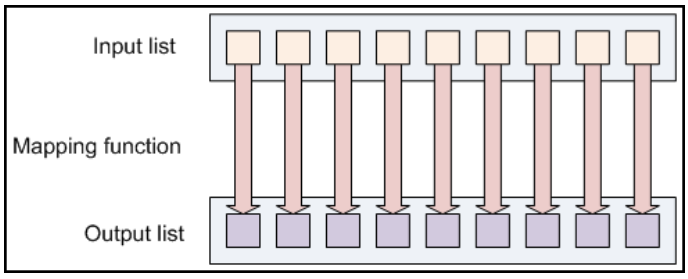
- Homework for **Week 2** – Thursday, 16 March:
- Given:
  - Example data set: 'MovieLens' data set about user-film ratings
  - Example Python/Java code of a file scanner in Map/Reduce fashion
- Task: Complete the given skeleton code so that it computes the average movie rating for films from a given range.
- The aim is to assess your programming skills
  - Understanding and working with a given processing framework
  - Functional thinking: Writing code for functions which solve a well-defined task in the context of a larger framework (in this case: filtering and aggregation)
  - Java and Unix skills (compiling, debugging)
- Detailed **Handout published on Piazza and eLearning**

Homework Tip 1: What is MapReduce?

- The Skeleton code implements a MapReduce simulator
- **MapReduce** is the main functional programming paradigm which we will use a lot throughout this class
- its main idea is that stateless functions are applied to individual elements of a list
  - a **map() (**or **filter())** function to transform some input element to an intermediate format; typical usage is data transformation and filtering
  - a **reduce()** function to fold a list of intermediate values to one result value; typical usage is aggregation
  - for a visualisation of this, see the figures on the following slide
- Please note that you only need to concentrate on those two functions, nothing else. All the program logic for reading the input, calling those functions, grouping intermediate results and passing around list of values is handled by a framework code which already exists.
  - later in the class, these will be frameworks such as Hadoop or Spark
  - in this homework, we implemented it already in the main program

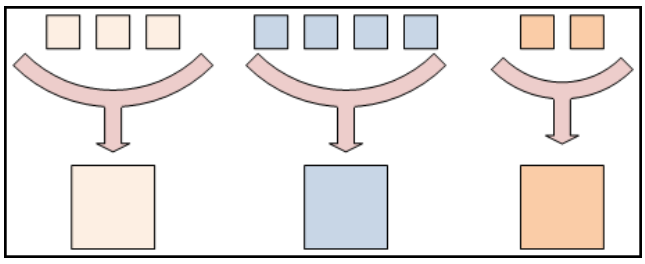
Figure 1: The MapReduce Model

map() (or **filter()**) function:
Input & Output: each a set of key/value pairs



Occasionally, map input key is used to associate map's input and output

Keys divide the reduce space
all of the output values are not usually reduced together. All of the values *with the same key* are presented to a single reducer together



reduce() function:
input a list of values for the same key,
output one value

Homework Tip 2: General Program Logic

The given skeleton code does quite a lot of work for you:

- it reads in a CSV file ("data.csv")
- it calls your **filter()** function for each line of this file
  - the filter function has to decide whether values in this line are processed or whether it is ignored
  - 'passing on' means returning a pair (or Record) of (key, value)
    - in our specific case, this should be (film_id, rating)
- The given skeleton code collects all filtered (film_id, rating) pairs, groups them by film_id, and then calls your **reduce()** function for each film_id with a list of the filtered ratings
  - your code should now return the average rating for a given film
- The skeleton code then prints out those results (plus some statistics about how many lines were passed or filtered)
- That's it!

Homework Tip 3: Input File Parsing

- Note that the given input file contains some incomplete data
  - like incomplete lines
  - or missing values
- Your code should check for any those problems
- Ignore any lines which are not of the expected format:
  user_id \t film_id \t rating \t timestamp
    - \t means a tab (data.csv is actually a tab-separated file)

Homework Tip 4a: Java Skeleton Code

- The Java skeleton consists of eight files:
  - **MapReduceSimulator.java** (and **FileScanner.java**)
    - main program, does not need to get changed; just take it as is
  - **Mapper.java, Filter.java, Reducer.java**
    - a series of abstract classes which you do not need to change
  - **Record.java**
    - defines a key-value pair; does not need to be changed, but you need to use it for the return value of your filter() function
  - **RatingFilter.java**
    - Here some work is needed:
      - a **constructor** for a RatingFilter object which keeps track of the range of film IDs which should be filtered
      - **the filter() method**; most implementation work should go here.
      - It is called for each line of the input file and should decide whether this is a film we are interested in, and if yes, return a Record with (film_id, rating)
  - **RatingReducer.java**
    - you only need to work on its **reduce() method**
    - This is called with a film_id and a list of all ratings for that film
    - Result should be the average rating for that film_id

Homework Tip 4b: Python Skeleton Code

- The second skeleton code is given in Python
- There are three files:
  - **map_reduce_simulator.py**
    - main program, does not need to get changed; just take it as is
  - **mapper.py**
    - defines a series of abstract and concrete Mapper classes
    - You only need to work on the **RatingFilter** class:
      - a **constructor** for a RatingFilter object which keeps track of the range of film IDs which should be filtered
      - **the filter() method**; most implementation work should go here.
      - It is called for each line of the input file and should decide whether this is a film we are interested in, and if yes, return a (film_id, rating) pair
  - **reducer.py**
    - defines a series of abstract and concrete Reducer classes
    - Only need to work on the **RatingReducer** class, its **reduce() method**
    - This is called with a film_id and a list of all ratings for that film
    - Result should be the average rating for that film_id