CSCI 5673 – Distributed Systems
Programming Assignment #1

Aditya Sawhney
4 March, 2011

Execute Hadoop MapReduce task on Amazon EC2 cluster

This is an auxiliary document along with the source code which details the approach and results.

Dataset

IMDB dataset (source http://warsteiner.db.cs.cmu.edu/db-site/Datasets/graphData/IMDB/) of all actors in the following format:

```
FIRSTNAME | LASTNAME | TITLE | YEAR | TYPE | POSITION | GENDER
```

The data set is around 213MB in size and contains 3.8 million records.

Problem

The idea is to use this dataset to compute a list of total movies each year and sort it in descending order. In addition, compute list of total movies per decade (binning) and visualize it in form of a histogram.

This is a bit tricky because the dataset corresponds to actors and not movies. We need to factor for the fact that:

1. There will be multiple records for movie-year combination as movies generally have more than one actor.
2. There will be multiple records with same movie name but different year as movie names can be reused.

Essentially, we need to *uniquely* count the movie-year combinations.

Approach

The approach is to break it down into 3 stages (corresponds to MapReduce) tasks and chain them together. Namely,

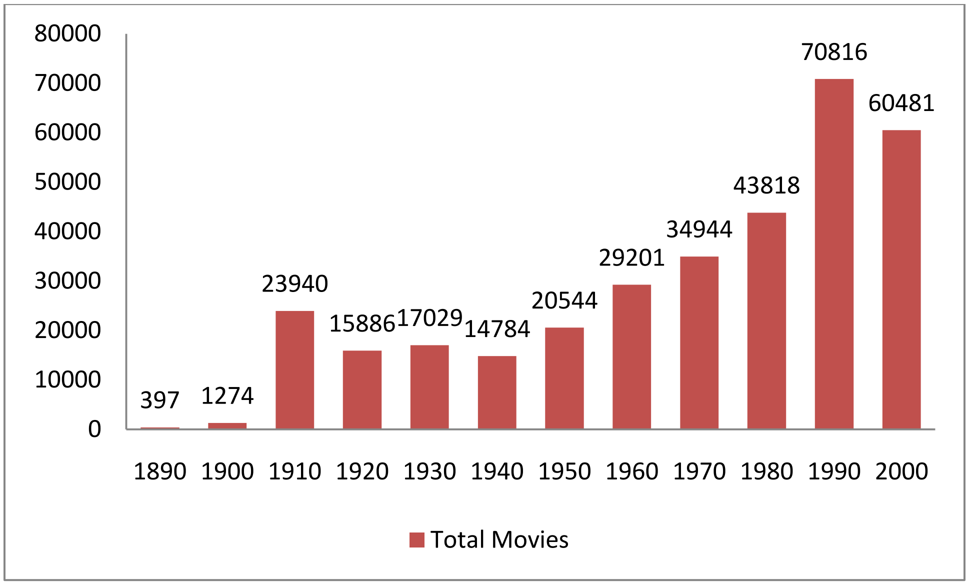
1. **Filter:** Filter out duplicate combinations of movie-year so that at the end we are left with unique combinations.
   a. **Map:** Parse each line of record, skip bad records and emit <Movie, Year>
   b. **Reduce:** Filter duplicate Year values for each movie and emit unique <Movie, Year>
2. **Count:** Count the number of movies each year
   a. **Map:** For each <Movie, Year> record, emit <Year, 1> and incase of 'histogram' round off year to start of decade.
   b. **Reduce:** Add up all the records for each year and emit <Year, Count>.
3. **Sort:** Sort the records by count in descending order and use single reducer to capture results in a single file.
   a. **Map:** For each <Year, Count>, emit <Count, Year> so that count is used as sorting key.
   b. **Reduce:** Use identity reducer as no change in map output is required and specify custom key comparator for output to sort in descending order.

Results

The results of normal and histogram (use -h option) are provided along with the source code. In terms of execution time following are the observations:

Phase	Normal (total movies/year)	Histogram (total movies/decade)
Filter	34 sec	37 sec
Count	19 sec	19 sec
Sort	20 sec	13 sec

The visualization of the total movies/decade in form of histogram is as follows:


🟥 Total Movies