NOVA SCHOOL OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Concurrency and Parallelism 2022-23 Lab 06

Hadoop Map-Reduce

Hervé Paulino & Alex Davidson

April 2023

Abstract

In this lab class you will acquire a better understanding of the MapReduce programming model and the Hadoop infrastructure to execute MapReduce jobs.

1 Introduction

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster made of commodity hardware. MapReduce is composed of a Map procedure that takes a slice of the input data and performs some transformation on that data, on the values themselves (e.g., multiply values by two) or on their structure/organization (e.g., filter words starting by vowels). The Reduce procedure takes the output from Map and applies some summary operation, which must be associative. The output of the Reduce procedure computation is the final result of the MapReduce computation. However, MapReduce computations may be chained in pipeline to produce more complex computations.

2 Task: Counting Word Occurrences

In this Lab work you are asked to setup a (single node) Hadoop MapReduce infrastructure, and then design some MapReduce programs to solve some simple data processing services.

2.1 Hadoop MapReduce using `jupyterlab`

This approach uses a free local instantiation of `jupyterlab`. First, go to `https://jupyter.org/install` and install it using `pip`.

The following steps will start a Jupyter Notebook that includes installing Haddoop MapReduce and then the job that will be executed.

1. Navigate to `https://classroom.github.com/a/oQ4WSom2` to create and clone your own repository containing two Python notebook applications: `mapreduce1.ipynb` and `mapreduce1_exercise.ipynb`.
2. Run the `jupyter-lab` application in this repository.

3. Open the "`mapreduce1.ipynb`" notebook and run it step-by-step. Observe and analyse what happens in each step.

4. Open the "`mapreduce1_exercise.ipynb`" notebook and solve the following exercises:

   - Write a MapReduce program that outputs the word frequency (output will be sorted by word).
   - Write a second MapReduce program that takes as input the output of (1) (i.e., the list of word frequency sorted by word) and will output the same list but sorted reversely (higher to lower) by the word frequency instead. As a suggestion, try to make a first version where the sorting is lower-to-higher, and afterwards attempt to sort higher-to-lower.

3 Task: Processing of Web Logs

Using the `mapreduce1_exercise.ipynb` as a *template* (you may duplicate and rename it to `mapreduce2_exercise.ipynb`), change it as necessary to:

1. Download the web log file from `https://www.dropbox.com/s/0r8902uj9yum7dg/web.log?dl=0` that contains a series of accesses to a particular URL path, the type of response, and the length of time taken to respond.

2. Create three MapReduce computations that processes the log of web entries to:

   (a) count the number of unique IP addresses included in the log;

   (b) for each interval of ten seconds, provides the following information: the number of requests, the average execution time, maximum execution time, minimum execution time;

   (c) create an inverted index that, for each interval of ten seconds, has a list of (unique) IPs executing accesses.

The biggest difficulty in the exercises above is how to collect the log entries that fit into the same *intervals of ten seconds* to produce the required information. Please use Piazza to discuss your ideas/strategy.