6.824 - Spring 2020

6.824 Lab 1: MapReduce

Due: Feb 14 23:59

Introduction

In this lab you'll build a MapReduce system. You'll implement a worker process that calls application Map and Reduce functions and handles reading and writing files, and a master process that hands out tasks to workers and copes with failed workers. You'll be building something similar to the MapReduce paper.

Collaboration Policy

You must write all the code you hand in for 6.824, except for code that we give you as part of assignments. You are not allowed to look at anyone else's solution, and you are not allowed to look at solutions from previous years. You may discuss the assignments with other students, but you may not look at or copy each others' code. The reason for this rule is that we believe you will learn the most by designing and implementing your lab solution yourself.

Please do not publish your code or make it available to current or future 6.824 students. `github.com` repositories are public by default, so please don't put your code there unless you make the repository private. You may find it convenient to use MIT's GitHub, but be sure to create a private repository.

Software

You'll implement this lab (and all the labs) in Go. The Go web site contains lots of tutorial information. We will grade your labs using Go version 1.13; you should use 1.13 too. You can check your Go version by running `go version`.

We recommend that you work on the labs on your own machine, so you can use the tools, text editors, etc. that you are already familiar with. Alternatively, you can work on the labs on Athena.

macOS

You can use Homebrew to install Go. After installing Homebrew, run `brew install go`.

Linux

Depending on your Linux distribution, you might be able to get an up-to-date version of Go from the package repository, e.g. by running `apt install golang`. Otherwise, you can manually install a binary from Go's website. First, make sure that you're running a 64-bit kernel (`uname -a` should mention "x86_64 GNU/Linux"), and then run:

```
$ wget -qO- https://dl.google.com/go/go1.13.6.linux-amd64.tar.gz | sudo tar xz -C /usr/local
```

You'll need to make sure `/usr/local/bin` is on your `PATH`.

Windows

The labs probably won't work directly on Windows. If you're feeling adventurous, you can try to get them running inside Windows Subsystem for Linux and following the Linux instructions above. Otherwise, you can fall back to Athena.

Athena

You can log into a public Athena host with `ssh {your kerberos}@athena.dialup.mit.edu`. Once you're logged in, to get Go 1.13, run:

```
$ setup ggo
```

Getting started

You'll fetch the initial lab software with git (a version control system). To learn more about git, look at the Pro Git book or the git user's manual. To fetch the 6.824 lab software:

```
$ git clone git://g.csail.mit.edu/6.824-golabs-2020 6.824
$ cd 6.824
$ ls
```

```
Makefile src
$
```

We supply you with a simple sequential mapreduce implementation in `src/main/mrsequential.go`. It runs the maps and reduces one at a time, in a single process. We also provide you with a couple of MapReduce applications: word-count in `mrapps/wc.go`, and a text indexer in `mrapps/indexer.go`. You can run word count sequentially as follows:

```
$ cd ~/6.824
$ cd src/main
$ go build -buildmode=plugin ../mrapps/wc.go
$ rm mr-out*
$ go run mrsequential.go wc.so pg*.txt
$ more mr-out-0
A 509
ABOUT 2
ACT 8
...
```

`mrsequential.go` leaves its output in the file `mr-out-0`. The input is from the text files named `pg-xxx.txt`.

Feel free to borrow code from `mrsequential.go`. You should also have a look at `mrapps/wc.go` to see what MapReduce application code looks like.