Data Acqusition

Jingci Wang

1/23/2019

This project aims at designing a system that automatically retrieves data from GitHub and generates datasets for future usage. The bash programming method I used depends much on what was introduced in lecture.

Framework

In my architecture, I defined a `git_log.sh` file to carry out the primary work. In order to collect targeted data from all the repository in `repos.list.txt`, I simply used a For-Loop to iterate through the entire list. Every time one repo fed into `git_log.sh`, three csv files would be created in the main directory. At the beginning of execution, the program would also ask for the file containing name-list of repositories so that it also works for a different set of repos.

For any given line from `repos.list.txt`, `git_log.sh` clones the repository to the local and set path into the repo directory by `cd`. Then it retrieves data using `git log` command. In the meanwhile, it writes the three kinds of targeted datasets into `${reponame}commit.csv`, `${reponame}message.csv` and `${reponame}file.csv` and move then back into the initial directory. With that, the path will also be set back. For space saving purpose, the fold would be immediately removed after extracting all the data from that repository.

Finally, all three kinds of dataset would be put together respectively

Due to the large scale of the dataset, it would take quite a long time to perform the entire process. Thus, I tried to parallelize it. I did it by first averagely splitting the `repos.list.txt` into 10 segments and execute 10 aforementioned For-loops with these segments of lists simultaneously. The code to implemented this is as follows.

```
for repos in xa* ; do {
for x in ` cat $repos `
{
user="$(cut -d'/' -f1 <<<"$x")"
repo="$(cut -d'/' -f2 <<<"$x")"
bash git_log.sh "$user" "$repo"
}
} & done
wait

echo "Done!"
```

I also used `date +%s` and `du -sh ${filename}` commmands to calculate running time and space taken by the output files.

Results and Discussion

Before parallelizing the process, the time for downloading the dataset is above 7 hours. After parallelism, the running time became around 4 hours. Since theoretically the running time should be expected to reduce 10 times over, parallelism did not actually work. When the data acqusition process finished, the shell gave result as follows.

File	Disk Space
commits.csv	3.9G
messages.csv	7.8G
files.csv	15G

| Running time | 13422 seconds |

API Method

At the beginning of this project, I tried to use python to scrapy the data with GitHub API. It worked with small test list. But when I feed the entire repos list into the program, it got stuck after several hours of running for more than once. And it was obvious that the data cannot be extracted within 24 hours, either. I have not found a certain reason for the program to be stopped but clearly it is not a good way due to the poor performance on time consumption. The python file is also included in the directory for reference.

Reference

Shell programming with bash: by example, by counter-example

BASH Programming - Introduction HOW-TO

How To Use Web APIs in Python 3