CSC326 - Programming Languages

Lab 4 Enhanced Search Engine

From Lab 1 to Lab 3, you have implemented a basic search engine, and deployed it on AWS. In this lab, you are asked to accomplish the following objectives:

- **Search Engine Enhancement**: Here you are given the complete freedom to design your own search engine.
- **Code Tidy Up:** Here you are expected to release your final project code externally, which requires you to clean up your code and scripts for understandability, maintainability, robustness and ease of deployment. *A substantial portion of your mark is based on overall code quality and style and documentation quality.*
- **Project Report**: Here you are required to write a project report to document your design, analysis and experiences.

Search Engine Enhancement

To enhance the search engine, you can either choose:

- ***algorithm route****:* to implement alternative implementations (hopefully better) of a selected key algorithm in the search engine;

  or

- ***feature route***: to add more innovative features at the frontend or/and backends.

Note that this part of the lab is open-ended, and you are *not* required to implement any
of the suggestions below. Whatever your choices are, you must describe your design decisions in the project report.

Examples of algorithm routes could be enhancing key algorithms with different computational complexities, or with different programming language paradigms.

Reference List of Features

Past examples of enhancements appropriate for the project are listed below. Note that you will likely need to implement **more than one feature** listed here.

Frontend

- Spell Correction
- Autocompletion
- Multi-word searching
- Search suggestion
- Query phrase interpretation, e.g. compute simple math equation if submitted as query.

Backend

- Complex Ranking System
- Store screenshots or summary for some of the highest ranked web pages
- Optimize search engine data structure, i.e. less storage, faster access, and etc.

Aesthetic and User Friendliness

- Minimize number of clicks for each search
  - *e.g. avoid clicking a button to go back the query page.*
- Customize results table for mobile devices, e.g tablets, smartphones, and etc.
- Animated logo, e.g. Google Doodles

Deployment and Autoscaling

- Running multiple bottle frontends and load balancer in one AWS instance to maximize resource utilization.
- Dynamically launching new bottle frontend by monitoring resource utilization.

Performance Improvement

- Optimize the number of connections that can be handled on the server, i.e 2000+ concurrent connections.
- Minimize processing time for each query, i.e. < 1ms. Note that the processing should exclude the network latency.

One-Click Deployment Script and Termination Script

In this Lab, you are required to submit a **deployment script** for launching your search engine on AWS, and a **termination script** for shutting down an active running instance from AWS.

Deployment Script

The deployment script does not only launch an AWS instance (like Lab 2), but also copy files to the server, and launches the frontend after it is started. This includes backend database files needed to demo the frontend. By the time when this script is completed, it should return an IP address or public DNS with port number (if not port 80), and your search engine has to be accessible from the public network.

A typical use of the deployment script is following:

1. User specifies AWS credentials in a separate key file;
2. User invokes your deployment script;
3. Deployment script loads the AWS key file, launches AWS instance, copies application files to the new instance, installs packages on AWS instance, and launch the search engine on server.
4. When the server is stable, the deployment script returns the IP address or public DNS of the new AWS instance. Also, the instance ID of new machine should be returned.
5. User accesses the search engine service through the returned IP address or public DNS from browser.

Note that if your search engine requires other 3rd party tools or API configurations, it is not required to include it in the deployment script. For example, if you are using Google Login APIs you can only specify the redirect URL and origin on Google Console GUI *after* the public DNS is returned from the new instance.

If it is genuinely impractical to include such configuration steps into the deployment scripts, then they are not required. Regardless the script must still deploy a **fully functional** version of your application to AWS. For example, the Google API must be configured manually ahead-of-time with a *static Elastic IP* (as suggested in Lab 2), so that your deployment script can setup your app to access the service *without having to configure it.*

Termination Script

The termination script is similar to the deployment script, except that instead of launching a new AWS instance, it shuts down an active instance. A typical use of the termination script is following:

1. User specifies AWS credentials in a separate key file;
2. User invokes termination script and pass the instance ID from command line;
3. Termination script shuts down AWS instance.
4. Upon completion, the termination script returns message indicating whether the termination process has been completed successfully.

Final Report

In the final report, you must clearly describe your search engine, and what features have been implemented in Lab 4. The final report should have the following sections:

1. Names and student numbers of all members.
2. Describe the design of your enhanced search engine in detail. If you enhanced an algorithm, describe the different candidate algorithms and how they are different from the baseline implementation in Lab 3, and describe the quantitative metrics (i.e. benchmarks, profiling tools) you use to judge the merits of the candidates and how you chose your final candidate.
3. **Brief** (no more than one page of text) high level documentation of your project's code, including where all features (i.e. *pagination is implemented in file_x.py , page ranking is implemented in files a.py and b.py*) are implemented, any external dependencies and how the different files relate to each other (i.e. high-level UML diagram).
4. Indicate the difference of your proposed design and completed design if there is any. If the search engine is completed differently than the proposed design, *explain why*.
5. Explain your testing strategy during the development. Describe how you identify the corner cases.
6. Lessons learned from this project.
7. Describe what you would do differently if you had to do it again. What would you do if you had more time. Did any parts take longer than you thought, and Why?
8. How the material from the course helped you with the project.
9. How much time it takes for you to complete each lab outside the lab sections.
10. Which part of the project you think is useful and you believe the labs should spend more time on it.
11. Which part of the project you think is useless and you think it should be removed from the labs when this course is being offered in the future.
12. Other feedback or recommendations for the course.

13. Responsibilities of each member. If you believe that workload is distributed unequally in your group, you may describe the situation in this section.

Deliverables

- Deployment script, termination script, all source files and a README file describing your code organization. **Note**: The README does not count as the report this time around, and should be a **concise**, ~1 page tutorial on how to run your code.
- Final report in **PDF**
    - final report should be named with your group number with following format: *group<group_number>-csc326-2014.pdf*

Hints

- ~~Data files generated by the crawler should be excluded from the submission, however, you will need to store the data files on your AWS instance to provides results for your search engine.~~
- Please include database files in your submission such that when your service is deployed it already has search results ready to go. Keep crawler database files below 1MB; this should be enough to demonstrate behaviour.
- The deployment and termination script may use third party libraries, such as Boto and others. However, you cannot assume such libraries have been installed on the test machine. Therefore, your script is required to detect whether the required libraries have been installed; if libraries are not found from the system, your script should initiate the installation process of the libraries.
    - Note that if you use a regular public AMI image for your AWS instance, the base operating system configuration is known in advance and dynamic detection of installation state *is **not** needed*.
- You may implement the deployment scripts in Bash or Python based on your preference. You may use multiple files to implement the deployment script. *However, the user should only be required to invoke **one** script.*
- When copying files from your own machine to AWS machines through SCP, the strict host key will be checked, which is not preferred for the deployment script. To bypass the host key checking, use "`scp -o StrictHostKeyChecking=no -i keyfile.pem local-file ubuntu@IP:~/remote-file`"
- For the termination script, you *may* require the user to enter a target IP address, instance ID, and etc, to identify which instance to terminate.
- When installing packages on AWS instance through 'apt-get' or other package managers, you may be required to answer 'yes' to continue the installation. To avoid such prompt, always append '-y' option to "apt-get" to enforce yes by default.

Submission

Compress all your files, including the source code *and the report*, and name it **lab4_group_<group_number>.tar.gz** with tar. Do not submit anything by email.

For example, for group 4, use the following command.

```
$ tar -zcvf lab4_group_4.tar.gz <all files>
```

Remember to decompress your archive before submission to check that all files are included. Empty archive submissions will be treated as late submissions. All submissions should come from one member in the group, *including resubmissions.*

To submit your package, use the following command on EECG machine. Note that the submission command must have the *-lab* prefix or you will submit an assignment instead!

```
$ submitcsc326f-lab 4 lab4_group_<group_number>.tar.gz
```