A Cloud-based Educational Competitive Programming Website

Bao Nguyen
Electrical Engineering and Computer Science
Vanderbilt University
Nashville, USA
cong.quoc.bao.nguyen@vanderbilt.edu

Blake Quigley
Electrical Engineering and Computer Science
Vanderbilt University
Nashville, USA
blake.d.quigley@vanderbilt.edu

Leqiang Wang
Electrical Engineering and Computer Science
Vanderbilt University
Nashville, USA
leqiang.wang@vanderbilt.edu

***Abstract*— Worldwide, the demand for technical jobs continues to grow, yet a variety of barriers to entry, ranging from a psychologically steep learning curve to physically expensive computer hardware, dissuade students from pursuing Computer Science. The team developed a website to make programming entertaining and easily accessible. In a classroom environment, students enter the teacher's generated class ID, complete an appropriate programming problem, and receive the output from their code executed in the cloud. To promote competitive drive, students are then ranked by various statistics such as time to first successful execution, average run-time, fewest lines of code, etc.**

Compared to similar solutions such as LeetCode, our website emphasizes classroom engagement and group competition. Additionally, compared to a traditional classroom environment, the cloud enables fast, easily joinable sessions without the need for every student to have a laptop and take time to setup a complicated, confusing programming environment. The planned design runs on Amazon Web Services in the cloud using Docker Swarm to orchestrate tasks among Docker workers executing the user code. Ideally, the solution would be used in situations such as STEM summer camps, club interest meetings, AP Computer Science classes, interview prep, technology company on-campus visits, etc.

Keywords—cloud, education, AWS, Docker, Flask, Python, Java, C++

I. Introduction

In the United States, companies have an ever-growing 918,000 unfilled IT jobs, far outpacing the 60,300 students graduating with a computer science degree every year [1]. Similar numbers persist worldwide. Arguably, the technical skills can be taught to most people; it is the passion and interest which is lacking. Significant barriers to entry hinder student interest in technology careers. For example, for financial reasons, students may not have consistent access to their own computer with administrator privileges. Even if they do, setting up a programming environment from scratch, following inconsistent online instructions to setup the PATH variable and so forth, may prove too much for students with less prior technical experience. Mentally, there is a steep learning curve between a simple "Hello World" program and more rewarding programming projects for youth such as website or videogame development. This learning curve may discourage them from even trying to start. In designing this website, the team sought to make computer science education fun and easily accessible.

A. Goals

To accomplish the goal of simple access, students join a class session by simply entering the class ID provided by their teacher, similar to Kahoot. For educational flexibility, teachers can choose from a variety of problems with different topics and difficulty, such as sorting an array of 100 integers. For reduced physical barriers, student code is executed in the cloud on powerful hardware which has already been set up appropriately. Students can also easily switch between a selection of programming languages with a simple browser dropdown instead of manual installation, or the teacher can require a specific language. To promote competitive drive, student solutions are ranked by the first successful execution, fastest average run-time, fewest lines of code or characters, etc. These results appear in the browser or, say, on the teacher's projector.

II. Design

In creating the website, a number of important design decisions were made in order to best accomplish the aforementioned goals. First, a unique game pin was chosen instead of requiring that students and teachers create and link individual user accounts. This decision was made to both enforce the easily accessible nature of the website, say for a 15-minute interest meeting, and to focus on the core website features, not user authentication and security problems. Second, several program languages are offered, namely Python, Java, and C++. More than one language is offered, despite the added complexity, to help users gain experience in multiple languages without installing and setting up multiple development environments. These specific languages were chosen due to personal experience, frequent usage in academia, and widespread workforce adoption. Python, Java, and C++ are used by 41.7%, 41.1%, and 23.5% of developers worldwide, respectively, according to Stack Overflow's 2019 developer survey [2]. Finally, the ranked statistics include first successful execution to promote low development turnaround time, fewest lines of code to promote creativity and competitive drive, and fastest average run-time to promote fast, efficient code.

III. Implementation

A. Front End

To fully implement the website, a number of front end and back end technologies were required. On the front end, a simple blend of HTML, CSS, and JS created a workable prototype to test the concept. To accommodate the project deadline, the team focused on producing a minimum viable product to demo, but, if this project were to continue long-term, standard industry web development resources such as React or Angular could help to manage the growing website. For user friendliness, the Ace code editor, with syntax highlighting, automatic indenting, etc. and support for over 120 languages, was employed. The micro web framework Flask serves these files and was chosen for accelerated development time and Python integration, which was the first language the site supported. Fig. 1 demonstrates the chosen layout and dark color scheme.

Fig. 1. Early iteration of front end design featuring IDE-like code editing features, language selection, and output

B. Back End

The design can be visualized in Fig. 2. On the back end, Amazon Web Services (AWS) performs all functions including the webserver and code execution. AWS was chosen for free

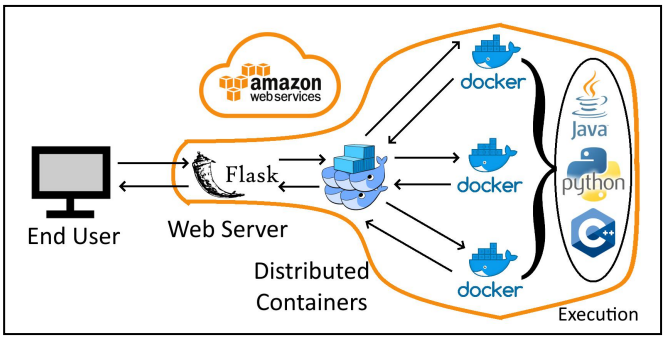

Fig. 2. Cloud-based design of website back end

education credits and to gain experience due to widespread industry use. To accommodate scaling website load, Docker Swarm orchestrates tasks among Docker workers actually executing the user's code. One advantage of the container-based approach is automatically handling some of the security concerns inherent with online arbitrary code execution through isolation. The workers execute the code, returning the result and run-time by finding the difference in start and end system time. A time out signal prevents the hogging of limited system resources (through an infinite loop or otherwise). Exception handling is also necessary to accommodate errors in user code.

For more technical details in our development, see the 'website' folder of our repository https://github.com/blakedq/4287-Final/tree/master/website

IV. Collaboration

In order to complete the project in a timely manner, the work was split and managed between the three group members. Nguyen helped with the worker nodes to communicate results and general back end support. Quigley was primarily responsible for the project idea, proposal, front-end design and implementation with HTML/CSS/JS, PowerPoint presentation, senior design day submission, and final report. Wang contributed to the master node for coordinating code execution tasks, web server using Flask, and integration with AWS. Additionally, all team members contributed to their relevant PowerPoint section, presented it in-class, and later demoed the website to a TA. For the purposes of this project, GitHub for file-sharing and GroupMe for communication proved sufficient. However, for larger long-term work, the collaborative development could warrant using tools such as CircleCI. To guide development efforts from a Minimum Viable Product (MVP) to a complete release, the team followed the stages in Table 1.

TABLE I. Development Milestone Goals

Stage	Goals
1 (MVP)	Create local webserver which can read user input from browser, execute Python, and return stdout.
2 (Expected)	Host webserver and Docker design in AWS. Provide user with problem and check output for correctness. Return statistics.

Stage	Goals
3 (Stretch)	Complete easily joinable competitive game, ready for classroom use. Include support for multiple languages and problems.

V. Conclusion

A. Cloud Computing Advantages

Many of the advantages discussed in the cloud computing class are fully realized in this website. For example, there was no need to acquire and setup physical hardware, removing huge capital costs and time delays. The cloud also largely removes many other complicating factors including maintenance, cooling, physical security, electricity, etc. Additionally, if the website were to see a sudden spike in usage, Amazon's vast server availability could accommodate, before scaling back down. At the same time, Amazon would only charge for the actual current resource usage. Also, the appropriate up-to-date environments for Python, Java, etc. only have to be set up one time in the cloud, instead of downloading and installing on individual user machines.

B. Limitations

The work was limited by the available development time and distraction of other school projects. Additionally, several potentially useful technologies such as Kubernetes and React were avoided because all group members lacked prior experience. For future work, the Docker workers should have pre-generated test cases for certain problems and compare the user's code output. Also, heavy system load should be addressed by proper scheduling to worker nodes. For a growing number of problems and test cases, a virtual machine for a database server could prove necessary. To inhibit malicious actors, the submitted code may need to be checked for security concerns such as system calls, e.g. disallow import of os in Python. Finally, more languages should be included to expand educational scope.

C. Summary

Overall, the project was relatively successful at creating a demo-ready proof of concept of the project proposal. The webserver is correctly configured on AWS, accessible from external machines, and capable of executing user code through Docker Swarm. Some components of the educational angle were not fully explored given the finite development time. For example, teacher/student user accounts could be created and linked to track participation or quiz grades. Also, the team would like to expand on the visually exciting competitive videogame-like aspects to further student motivation and passion. All of the code as well as this report can be found at https://github.com/blakedq/4287-Final.

Acknowledgment

The team wishes to thank Professor Aniruddha Gokhale for the teaching received in the Vanderbilt course CS-4287 Cloud Computing which inspired the website's creation and guided the technical design and implementation.

References

[1] J. Liu, "The US has nearly 1 million open IT jobs-here's how much it can pay off to switch industries into tech," CNBC, 06-Nov-2019. [Online]. Available: https://www.cnbc.com/2019/11/06/how-switching-careers-to-tech-could-solve-the-us-talent-shortage.html. [Accessed: 12-Dec-2019].

[2] "Stack Overflow Developer Survey 2019," Stack Overflow, 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019#most-popular-technologies. [Accessed: 12-Dec-2019].