# Project Report: Distributed Shell For Grading Homeworks

**Team Name:** dsh
Aidai Beishekeeva, ab5248
Yaxin Chen, yc3995

## 1. Synopsis

In this project, we built an auto-grader with two distributed shells, pssh and dish, and compared these two shells by their efficiency. The auto-grader works for scenarios in some programming courses where each student is assigned a virtual machine to work on their assignments. Our tool is able to help the instructor provision these virtual machines by facilitating environment setup for assignments and parallelizing the grading process on each student machine.

The prospective user community for which our project is targeted is teaching teams where the coursework requires programming homeworks and users of distributed shells. Assignments can be of varying complexity as long as they can be auto-graded with the help of the script. This will free up time for instructors by using automated and distributed nature of the setup we described.

## 2. Research Questions & Methodology

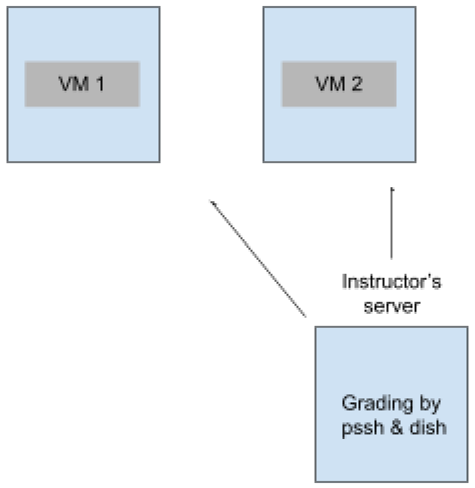
After setting up both of the environments, we will answer the following question:

RQ1: How easy is it to set up the environment and configuration for each shell to be able to grade homework on distributed systems?

RQ2: Which shell is better according to the execution time and the ability to handle connection failures?

## Methodology

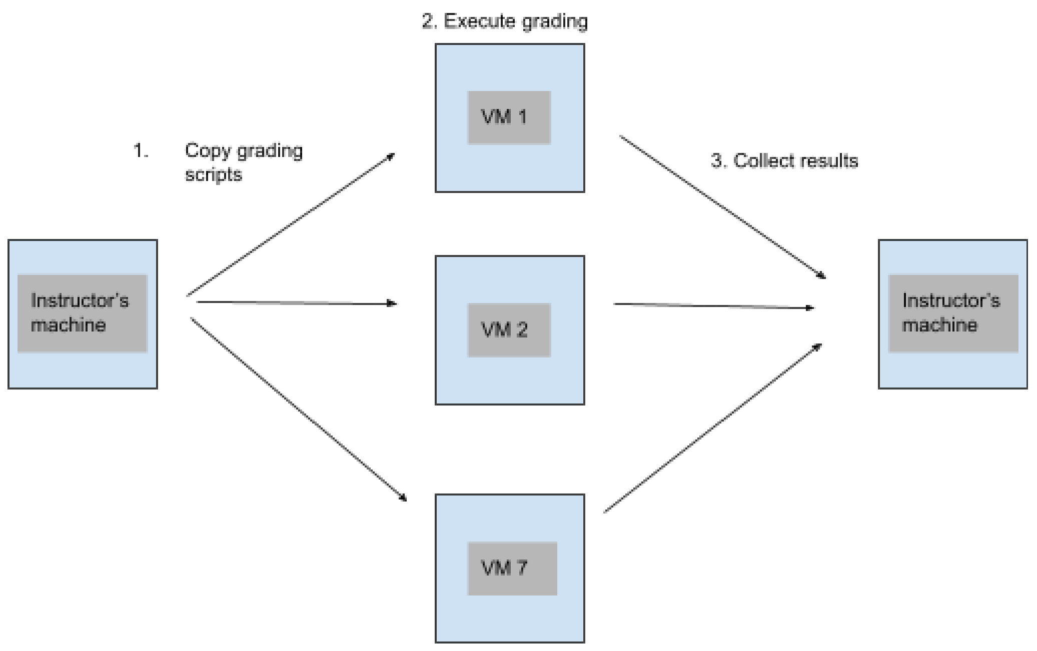
To compare the work of the two shells, each of us set up an environment in Google Cloud consisting of 8 virtual machines: 1 for instructor, 7 - for students, as depicted in the following diagram:



Each virtual machine runs Debian 10 with 2vCPU and 1GB memory.  Each student machine has a connection with instructor's server via SSH keys.

To use our tool, the instructor only needs to write a configuration file besides the grading script for one student submission on a single machine. The configuration file should include the user names and ip addresses of all students' machines, and other optional fields such as the path to the grading script and the path to store the grading result. If specified, our tool will first copy the local file or directory, which usually contains the grading script or assignment starter code, from the instructor machine to all the students' machines. Then, it will execute the commands specified by users that need to be executed before grading, which can be commands for package installation. Next, it will run the grading script under the grading directory on each student's machine, collect the grading results and store them on the instructor's machine. Finally, it will execute the commands that need to be executed after grading, which can be commands to remove the grading directory.

The grading flow is depicted here:



All of the three steps above are completed when you call the following command on both environments.

```
python3 grading.py config.json
```

The purpose of `grading.py` script is to parse `config.json` and call the respective commands for `dish` and `pssh` that in their turn perform the copying of the files, execution of grading and saving the results.
The example of `grading.py` can be found here:
https://github.com/yaxinchen666/COMS6156-distributed-shell/blob/master/pssh/grading.py

The simplest `config.json` can look like this:

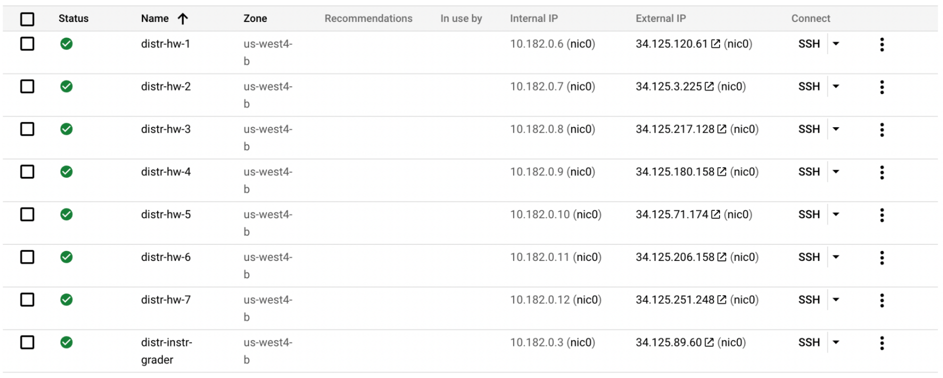
```
{
"time out" : 30,                           # timeout for the command above
"instructor" : "aidai_beishekeeva",        # needed to login on student machine(SM)
"host_file" : "host_file",                 # list of usernames & ips of SM
"local_file" : "hw1_grader.py",            # grader file with unit tests
"destination" : "~",                       # directory on SM
"grading_script" : "hw1_grader.py",        # file name for grading script
"grading_dir" : "~",                       # directory for grading
"pre_commands" : [],                       # commands to run before grading
"post_commands" : [“rm -rf hw1_grader.py”],# commands to run after grading
"result_path" : "hw1_grading_result",      # file with results on instructor’s VM
}
```

Both of the distributed shells use the underlying `scp` command on linux machines to copy the grading files. The files are copied to destination directories specified in the configuration file.

Configuration file also contains the path for grader file that is called to execute, for example, different test cases against the user’s solution files. For both `dish` and `pssh`, we created a simple python file that uses unittest library and imports the student’s solution file from grading directory (specified in config.json)

Pre commands are run if there is something that needs to be run before the grading script. For example, installing a library with `pip install numpy`. Post commands are run after the grading is finished. In our case, the post command we are using is removing the grader file containing the unit tests.
Finally, after the grading is finished, the results of unit tests are saved in the file specified in `config.json`.

The setup on Google Cloud looks the following way:



| ☐ | Status | Name ↑ | Zone | Recommendations | In use by | Internal IP | External IP | Connect | |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | ✓ | distr-hw-1 | us-west4-b | | | 10.182.0.6 (nic0) | 34.125.120.61 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-2 | us-west4-b | | | 10.182.0.7 (nic0) | 34.125.3.225 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-3 | us-west4-b | | | 10.182.0.8 (nic0) | 34.125.217.128 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-4 | us-west4-b | | | 10.182.0.9 (nic0) | 34.125.180.158 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-5 | us-west4-b | | | 10.182.0.10 (nic0) | 34.125.71.174 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-6 | us-west4-b | | | 10.182.0.11 (nic0) | 34.125.206.158 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-hw-7 | us-west4-b | | | 10.182.0.12 (nic0) | 34.125.251.248 (nic0) | SSH ▾ | ⋮ |
| ☐ | ✓ | distr-instr-grader | us-west4-b | | | 10.182.0.3 (nic0) | 34.125.89.60 (nic0) | SSH ▾ | ⋮ |

## Results and Findings

We generated text files of size 1MB, 10MB, and 100MB and tested the time it takes to copy the files from the instructor machine to the 7 student machines. The result is shown in the following table:

| | PSSH | DISH |
|---|---|---|
| Time to copy a file (1MB) | 0.778s | 0.872s |
| Time to copy a file (10MB) | 2.874s | 1.247s |
| Time to copy a file (100MB) | 94.753s | 5.852s |
| Min number of files for environment setup | 4 | 4 |

Table 1: Time comparison for copying file of different sizes on students' machines

We created a grading script https://github.com/yaxinchen666/COMS6156-distributed-shell/blob/master/test/grader.py, which takes 2.573 seconds to run locally. We tested the time for distributed grading on multiple student machines.

| | PSSH | DISH |
|---|---|---|
| Time to grade on 1 student machine | 5.131s | 5.840s |
| Time to grade on 7 student machines | 5.411s | 5.908s |

Table 2: Time comparison for overall execution of grading script

### RQ1: How easy is it to set up the environment and configuration for each shell to be able to grade homework on distributed systems?

Both pssh and dish are based on ssh and require the same 4 files to set up the overall grading environment: grading.py, hw1_grader.py (containing the unit tests), host file, config.json. Once ssh connection is established, pssh and dish can be used directly.

### RQ2: Which shell is better according to the execution time and the ability to handle connection failures?

Based on the two tables above, the execution time of the grading script is roughly the same for both distributed shells given that the environment setup is identical, but the DISH is faster than PSSH on transferring large files. The execution time of grading on 1 student machine is roughly the same as the time of grading on 7 student machines, which indicates the grading is in parallel. For DISH, the default timeout for distributed command is 30 seconds if the instructor's machine is not able to login into of the student's machines; for PSSH, there is no timeout by default, which means it keeps trying to connect until success. Both DISH and PSSH are able to handle temporary connection failures within timeout.

To sum up the comparison of the two shells, it is fair to say that both of the tools performed equally well in terms of copying files and executing the grading. Because of the limit on virtual machines in Google Cloud, we were only able to test the distributed shells on 7 machines. Future work includes setting up and testing the environment that includes few hundreds virtual machines. According to the documentation of DISH "it shouldn't be a problem to process a few hundreds of hosts in parallel" [1].

## 3. Deliverables

Link to repository: https://github.com/yaxinchen666/COMS6156-distributed-shell

## 4. Self-Evaluation

Aidai: Implementing this project made me aware of not only the workings of distributed shells, but also of their impacts in educational domain. Based on my experience at Columbia, many computer science courses have at least ~100 students. These courses can benefit instructors in grading the assignments, saving time on other important aspects of teaching and provide better and timely feedback to students on their homework.
While working on this project and testing out the DISH, I witnessed how the use of distributed shells simplified many tasks and I can only imagine the power and use it brings in large scale distributed systems in industry.

Yaxin: I was responsible for building and testing the tool with PSSH. The most challenging part for me was building a cluster on google cloud platform and designing the functions and workflow of the grader. Connection via ssh is a little bit tricky on google cloud platform, and Aidai and I spent quite a long time to figure out that we need to specify username and expire time when generating and adding SSH key in order to avoid 'permission denied' error. As for the design, we brainstormed the functions and the configuration of our grader, and then I formalized its workflow. Through this project, I learned how to build a tool using distributed shell, and that distributed shell can be a very powerful tool for cluster management.

## 5. References

1. dish: https://freeshell.de/~drimiks/gnu/download.cgi/progs/dish/dish.1.html
2. pssh: http://manpages.ubuntu.com/manpages/focal/man1/parallel-ssh.1.html