COMP90015 Distributed Systems Project2 Report

Tanqi Zhao tanqiz 710689
Yannian Liu yannianl 706653
Zheng Tian ztian2 710453

1 Introduction

In this project, we intend to build a job management system, which implements a worker-master architecture. Workers and masters are in many-to-many relationship. Master can send a job (jar file) and a input file to the worker to run. The worker deal with the job and send back the output file to the master. Communication between the worker and the master is through SSL encryption to ensure security. Nectar cloud service provides remote VM service as the worker.

2 Architectural model

In this project, a Master-Worker model is applied as shown in the figure. The system mainly contains two parts: master and worker. From the master, end users can connect to the worker and send a job to the worker to deal with. A job consists of a jar file and a input file. After receiving the job, a worker starts a thread to deal with the job. When the job is finished, our worker sends the output file back to the master, so the end-user receives the corresponding output file of the job. Meanwhile, users should be able to monitor the status of their jobs (submitted, processing, failed or successful) and workers.

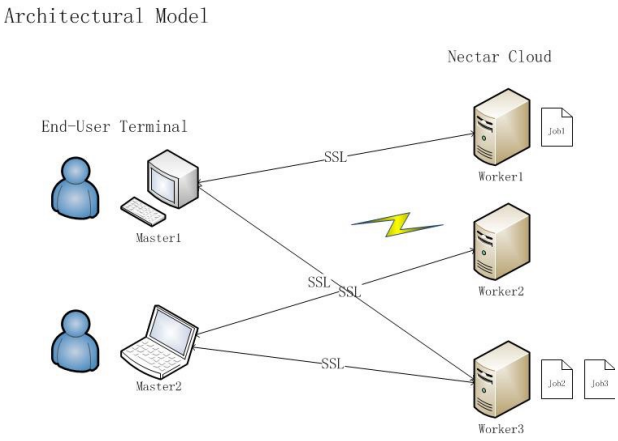

FIGURE 1. ARCHITECTURAL MODEL

Masters could be implemented on any end-user's machines. Different masters can submit their jobs to workers regardless of each master or worker's status. Also, new workers could be added by defining an address and a port.

3 Implement

3.1 Communication implementation

The means of communication between the worker and the master is through SSL(Secure Socket Layer), which makes sure network security between the two nodes.

The Secure Socket Layer protocol is intended to provide a flexible means for clients and server to communicate using a secure channel. SSL encrypts the network connection in network layer. SSL layer is between TCP layer and application layer. Data from TCP layer no longer directly go up to transport layer but to SSL layer for encryption, and being added a SSL header itself.

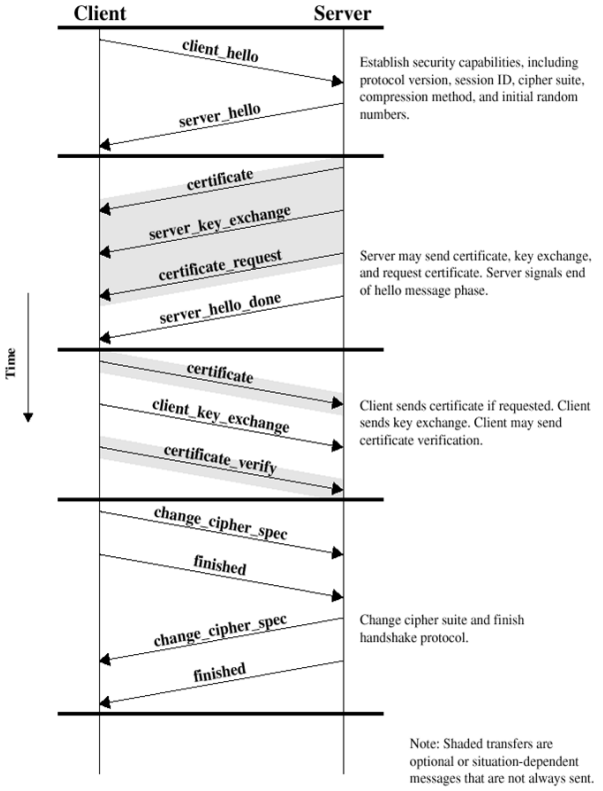

FIGURE 2. 4 PHASES OF HANDSHAKE PROTOCOL

Mechanism for SSL: 1. Handshake protocol, 2.Record protocol, 3. Content protocol

There are three basic phases:
1.Peer negotiation for encryption and authentication algorithm support.
2.Public key encryption-based key exchange and certificate-based authentication.
3.Symmetric cipher-based traffic encryption.

The first phase allows the client and server to establish which cipher, compression method and other connection parameters are to be used.

The second phase exchanges certificates. A master secret or common secret is negotiated and this is used to generate all other key data for the purpose of encryption.

3.2 Protocols and Technologies

We used TCP (Transmission Control Protocol) and SSH (Secure Shell) in our project.

3.2.1 TCP

The main reason why we choose TCP is that in this project master and worker need to establish a connection before transferring files and TCP meets this requirement. TCP provides a connection-oriented, reliable byte stream service. Connection-oriented means the two applications which are using TCP (usually a client and a server) must establish a TCP connection at first before exchanging packets to each other. Besides, in TCP connection, the data stream must be served on the other in the correct order.

3.2.2 SSH

SSH is reliable, which is a protocol designed to provide security for remote login session and other network services. SSH protocol can effectively prevent information disclosure in the use of remote management.

From the client perspective, SSH provides two levels of security authentication. The first level is password-based security authentication and the second level is key-based security authentication.

In the first level, users can log in to the remote host as long as they know their account numbers and passwords. All transmitted data will be encrypted, but It can not guarantee that the server users are connecting is that one which users want to connect. There may be other servers in posing as the real server. It means the connection is under attack by the method of “Middleman”.

The second level need to rely on keys. Namely, user has to create a pair of keys and put a public key on the server which needs to access. If user wants to connect to the SSH server, the software on the client will send a request of using the keys for security verification to the server. After the server receiving the request, the server find the user's public key on the server under the user's home directory at first. Then the server compare it with the public key user has send. If the two keys are the same, server encrypt “challenge” with the public key and sends it to the client software. After receiving the “challenge”, client software can decrypt it using user’s private key and then send it back to the server.

This project use the second level, key-based security authentication. In this way, you must know the password key. However, compared with the first level, the second level don not need to send passwords over the network. Besides, The second level not only encrypt all data transmitted, but also protect transmission from “middleman" (because “middleman” do not has user’s private key).

3.3 UML Sequence diagram

UML Sequence Diagram

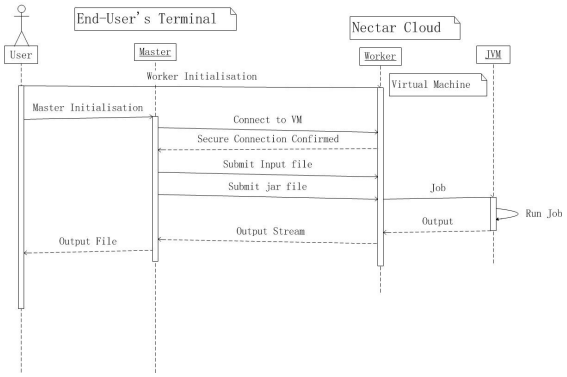

FIGURE 3. UML SEQUENCE DIAGRAM

In this system our group use WinSCP for Windows and Cyberduck for Mac OS X to transfer files to the virtual machines of the cloud, which is more convenient. While during the communication of the 2 nodes, SSL is used within our java code.

4 Threads and processes

4.1 Threads

This project establishes a thread when the connection between master and worker is established. It is multithreading because the master maybe connect with more than one workers.

Besides, a thread will be built when user clicks the "sent" button. In each thread, the master starts to sent a jar file and an input file, monitor the status of one job and receive the output file. The reason why multithreading is used over here is that the master need to sent several files to worker, receive several files from worker simultaneously and monitor the status of all of jobs simultaneously.

4.2 Process

When worker receive a jar file and an input file, a process will be established for run the jar file with the input file and output a result file. If the thread which is running file has some problems, its error does not affect the threads.

5 Failure Handling

Java provides a try-throw-catch mechanism for handling exceptions. In the process of system implementation, all kinds of failures may occur.

Network failure: unable to connect to the virtual machine due to connection problem or service problem. Solution: return corresponding information and try to reconnect.

Job failure: unable to implement the submitted job, because of incorrect file format or all kinds of exceptions. Solution: inform the master, return corresponding error message.

VM failure: failed or crushed virtual machine, unable to execute commands. Solution: inform the master, reboot the VM.

6 Assignment of each team member

Assignment of Each Member

FUNCTIONS	Zheng Tian	Yannian Liu	Tanqi Zhao
MASTER			
Accept a list of addresses and ports		√	
Connect workers by SSL connection.	√		
Algorithm Round Robin of selecting a worker	√	√	
Sending a pair of jar file and input file to a	√	√	
Monitor the status of an already submitted jobs, e.g. RUNNING, FINISHED, DISCONNECTED or FAILED.	√	√	
Receive the output file or report the error	√	√	
Monitors and displays all workers nodes and their status	√	√	
Add new worker dynamically	√	√	
WORKER			
Accept Multiple jobs from multiple master	√		√
Create a process to run the jar file after receiving a pair of a jar file and a input file	√		√
Report status of jobs to master	√		√
deploy each worker to its VM as a runnable jar file over SSH/SCP			√
Return the output file to master	√		√
OTHERS			
Graphical User Interface (GUI)		√	
Allow the user to specify a deadline for each job. If the job does not finish within the deadline then terminate the process and mark the job as failed	√	√	√
Report		√	√

7 Conclusion

In this project, our group have carried out most of the project spec has asked through java coding and environment deployment. A job can be sent to a remote VM for execution and user can get corresponding output file according the input file. In the meantime, the masters can monitor the status of works and jobs the submitted. Our system can basically show some ideas of distributed systems, from which we have learned a lot.

However, due to lack of experience and some other reasons, out project has much to improve, and that is exactly what we intend to to in the future research and study.