COMP3100 Group Project

## Project title

Intelligent cloud storage distributed system (ICSDS)

## Introduction

With the rapid development of science and technology in recent decades, especially the popularity of the Internet and the comprehensive construction of network infrastructure, humans have become increasingly dependent on network applications in the daily life and work job, and intelligent cloud storage applications are among the fastest-growing.

## Aims

Our goal is to enable customers to upload/share/transfer data through our ICSDS designed between different locations, devices, and even different OS, so that users can operate seamlessly data between multiple devices.

## Background

### Architecture

The architecture diagram of the ICSDS is shown in Figure 1. It separates the application services, data services, and log services. When one system service changes in the future, it will not affect other service systems in normal operation. At the same time, it is also convenient when upgrading one group of servers. Application services serve as interfaces that are directly connected to users and will access corresponding services.

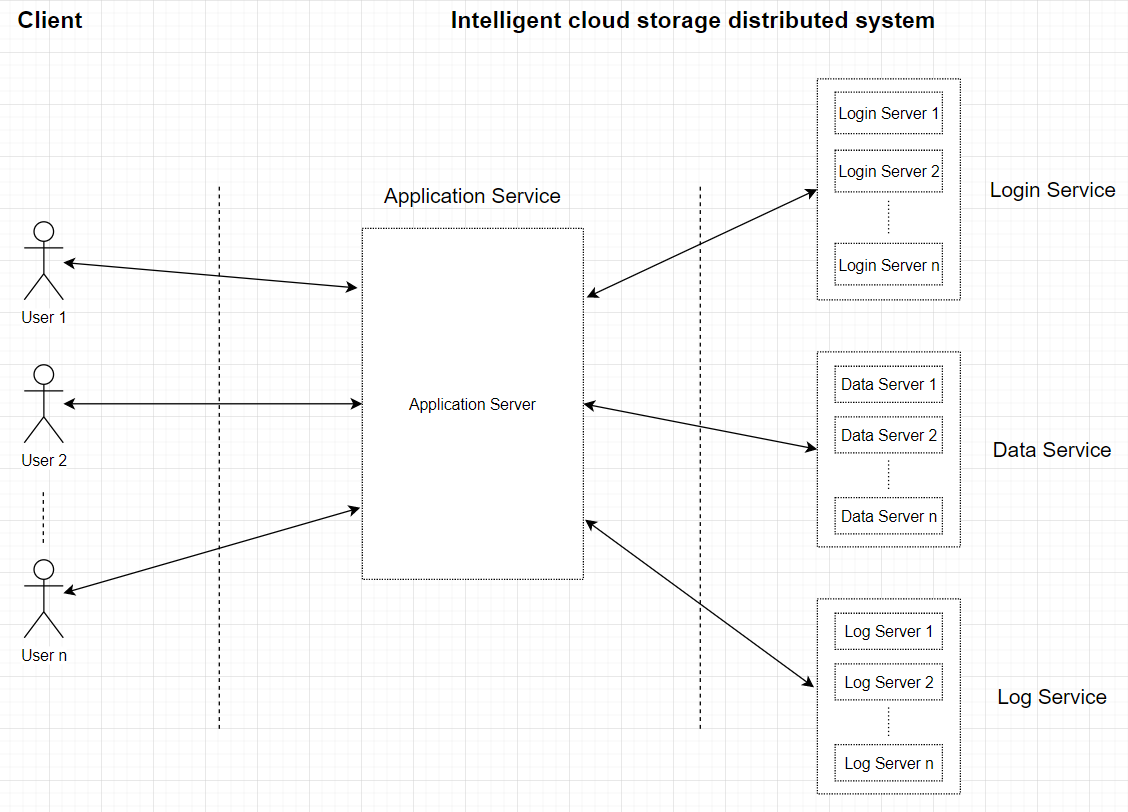


Figure 1 ICSDS architecture

### Discrete even simulation

**Initialise** system state variables by reading

• Simulation configuration file (e.g., ds-server-config.xml) in the server-side simulator

• Simulation configuration file (e.g., ds-client-config.xml) in the client-side simulator.

Client **sign in**

* Login verification

**Insert a log** for user login

Client **choosing work mode**

* upload
* download

**While** all files are uploaded/downloaded **do**

• Insert a log when a file transfer completed

**End while**

Client **sign off**

**Insert a log** for user sign off

### Client-server communication model

ICSDS decided to use persistent and synchornous as the client-server communication model, because when users are transferring files, it needs to keep the communication persistent. Moreover, when the communication is interrupted, the server can retain the message which is sent unsuccessful and resend it when the communication is restored, rather than discarding the message directly.

## Project plan

### Project roles

|  |  |  |
| --- | --- | --- |
| Name | OneID | Roles |
| Xinglin Chen | 44089333 | Project Management & Testing |
| Fei Huang | 44129866 | Design Algorithm & Programming |
| Jiahui Lin | 45141916 | Design Algorithm & Programming |

### Schedule

|  |  |
| --- | --- |
| Tasks | Due Date |
| Project Design Document | 20/03/2020 |
|  |  |
|  |  |
|  |  |

### Work arrangements

### Data management

We put all the details in the GitHub which link as below:

<https://github.com/SnakeCN21/COMP3100-Group-Project>

### Programming language

We decided to choose Java as the programming language. First of all, it has a high readability, and secondly, it also has a wealth of libs to choose from to help us complete the development of the entire program.

## References

* Alexander Libman and Vladimir Gilbourd. “Comparing Two High-Performance I/O Design Patterns”. In: *Research Journal of Applied Sciences, Engineering and Technology* (Nov. 2005). online. url: https://www.artima.com/articles/io\_design\_patterns.html (visited on 19/03/2020).
* Barile, I. (2004). I/O multiplexing & scalable socket servers. *Dr. Dobb's Journal*, 29(2), 42-45.
* Fielding, R., & Taylor, Richard N. (2000). *Architectural Styles and the Design of Network -based Software Architectures*, ProQuest Dissertations and Theses.
* Gosling, J., Joy, Bill, & Steele, Guy L. (1996). *The Java language specification / James Gosling, Bill Joy, Guy Steele*. (The Java series). Reading, Mass.: Addison-Wesley.
* Mockford, K. (2004). Web Services Architecture. *BT Technology Journal*, 22(1), 19-26.