Operating Systems
Spring 2018

Operating Systems Project

Professor : Philippe Cudre-Mauroux
Assistant : Ines Arous

Submitted by Groupe 4: Sylvain Julmy, Michael Papinutto, Sami Veillard

---

May 25, 2018

Introduction

For this project, we had to implement a multi-threaded client-server system using TCP sockets. This system had abilities to write keys with values, read values providing a key, simultaneous safe access of the readers and the writers. This system was subsequently tested using an automated testing in form of a bash script.

Chosen approach

We have to take care of various aspect when choosing the data structure to store the key-value entry : maximal number of entry stored in the database, extension of the data structure, number of simultaneous access on the server and how to synchronize the threads.
A lock-free hash-set data structure offers solutions to all of those challenges. We have implements our own data structure in C based on the one created by Herlihy in [Her06] in Java.

Challenges encountered

One of the challenge we encountered was to read or deleted entries from values. Indeed, looking for a value in a hash-table is not as straight forward as it seems to. It required to navigate through all the keys until finding the corresponding value. Despite the high cost of this operation, we decided to use this method as we did not find any other way top provide such an operation.

Conclusion

In conclusion, we addressed the challenges proposed in this project using a lock-free hash-set data structure. Despite

Bibliography

[Her06] Maurice Herlihy. "The art of multiprocessor programming". In: *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing - PODC '06*. 2006. ISBN: 1595933840. DOI: 10.1145/1146381.1146382.

Documentation

Communication

The server and the clients communicate through socket.

Server

The server is composed of several binary files. The main file encompasses the socket setup for the server and dedicated files for communication and the shell graphical user interface. The server is multi-threaded, *i.e.*, after each connection of a client the server create a new thread. The data structure is described above.

Server usage

TCP Port	5000 (can be reset in the server main file)
.\server	server start

Client

The client is also composed from several binary files. The main file set up client socket and dedicated files are used for execution of command and shell graphical interface. On the contrary of the server shell, the client shell is an interactive shell which usage is described below. Moreover, to simplify benchmark the client can also accept files at launch.

Client usage

Client basic usage

.\client <server ip address>	client start
.\client -option <server ip address>	client start with options (see below)

Options at start

-? -h --help	client command help
-f <file> --file <file>	client start and execute command present in the file specified after this option
-F <file1> ... <fileN> --files <file1> ... <fileN>	client start and execute command present in the files specified after this option

Client command accepted in interactive GUI

add <value> or add <key> <value>	add a value to the database with or without generated key
ls	list the content of the database (unordered)
read_v <key>	read a value in the database from a key
read_k <value>	read a key in the database from a value
rm_v <key>	delete a value in the database from a key
rm_k <value>	delete a value in the database from a key
update_kv <value> <newvalue>	update an entry in the database