Journal of Computing Research and Innovation (JCRINN) Vol. 7 No. 2 (2022) (pp1-10)
https://jcrinn.com : eISSN: 2600-8793 doi: 10.24191/jcrinn.v7i2.271
https://doi.org/10.24191/jcrinn.v7i2.271

Parallel and Distributed Computation of a Fingerprint Access Control System

Bopatriciat Boluma Mangata[1]*, Kadima Muamba[2], Fundji Khalaba[3], Bukanga Christian Parfum[4], Kisiaka Mbambi[5]
[1,3,4] Faculty of Science and Technology, University of Kinshasa, Kinshasa, D.R.Congo
[2,5] Faculty of Computer Science, Reverend Kim University, Kinshasa, D.R.Congo

*Corresponding author: * bopatriciat.boluma@unikin.ac.cd*
Received Date: 25 February 2022
Accepted Date: 27 April 2022
Revised Date: 10 May 2022
Published Date: 1 September 2022

HIGHLIGHTS

- Design of a fingerprint-based access control system to secure premises.
- Evaluation of the execution time of a fingerprint-based access control system, sequential and parallel approach.
- Comparison of the execution time of a fingerprint-based access control system to secure premises, sequential and parallel approach.

ABSTRACT

This work evaluates the runtime performance of a single-mode biometric recognition system for fingerprint-based access control to secure premises. To speed up the computation time in this system, we resorted to parallel programming, targeting more loops in the verification module. Our approach would therefore be to parallelize all loops that are computationally intensive during the verification of fingerprints in the database. On this, we exploited Microsoft's Task Parallel Library, specifically exploiting the for and for each loop. On the test set performed in sequential and parallel versions in the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600, we can state that the results obtained by the sequential and parallel implementations of our performance test programs allowed us to determine the best approach. Therefore, it is very clear that the sequential program is too greedy in terms of computation time compared to the parallel program which minimizes the computation time.

Keywords: Task Parallel Library, Biometrics, Fingerprint, Access control, Parallel computing.

INTRODUCTION

Problem

Each technological progress opens the horizon to new needs. Applications are becoming very demanding in terms of computing time and memory space, especially real-time and simulation applications. Parallelism has always been a possibility to meet this demand for performance (Fryza, Svobodova, Adamec, Marsalek, & Prokopec, 2012).

The real problems that arise, as far as verification in a student access control system in an institution is concerned, especially when we have a mass of information in the database, are slow, high computation time, which makes verification tedious (Williams-Young, De Jong, Van Dam & Yang, 2020).

Thus, in order to motivate the continuation of this work, questions of the kind listed below will not be ruled out:

- ✓ What are the most efficient methods we can apply to reduce the computation time in an access control system?
- ✓ How can we optimise the runtime computation problem in a fingerprint-based access control system?

These issues are the real problems that we will examine in the following.

Assumptions

To solve these problems, the optimal solution we propose in this work is to exploit parallel programming, with the aim of improving the capabilities of the parallel computing verification module through the implementation of parallel loops.

More precisely, we will design a parallel computation verification module based on Microsoft's Task Parallel Library, exploiting more precisely the loop for and loops for each.

Objective

The general objective of this work is to design a tool that will optimize the computation time of an access control management system.

Interest of the subject

The interest of such an approach is to make a major contribution to the scientific community, by providing them with a logical approach to optimizing the runtime performance of an access control system for premises secured by fingerprints.

PARALLEL PROCESSING

Some general information

Parallel processing is a form of information processing that allows the exploitation of concurrent events at runtime. These events are located at several levels: at the program level, at the procedure level (coarse-grained parallelism), at the instruction block level (medium-grained parallelism) or within an instruction (fine-grained parallelism) (Reumont-Locke, 2015)

Parallelism is the fact of making several processors cooperate with the aim of accelerating the resolution of a single problem, improving computing performance, increasing the size of the problems to be solved, producing machines with a good cost/performance ratio (Tavara, Schliep, & Basu, 2021).

The introduction of parallelism within a program can be done at the level of procedures or even loops of the procedure (Ocaña, & de Oliveira, 2015). It requires the decomposition of the program into tasks, the search for dependency relationships between these tasks by constructing a directed graph, whose vertices represent the tasks and edges represent the dependencies between the tasks, called a "dependency graph", and the parallel programming of independent tasks (Abdellatif, M. (2016)).

Presentation of the work

This work is in the context of parallel application programming which requires high computational capacities. Its objective is to study the execution time performance of an access control system for premises secured by fingerprints (Miao, Tian, Peng, Hossain & Muhammad, 2017)

To do so, we proceed as follows (Bopatriciat Boluma Mangata et al., 2022):

- ✓ We take our fingerprint verification program, in its sequential performance test version, on a set of six hundred individuals.
- ✓ We run this sequential version on the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600. The aim here is to evaluate the execution times of each data size by responding to the sequential test program.
- ✓ We parallelize our test program, namely the fingerprint verification program, using the parallelism of the for and foreach loops of the Task Parallel Library.
- ✓ We run this parallel version on the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600. The aim here is to evaluate the execution times of each data size by responding to the parallel test program.
- ✓ We interpret the results obtained from the sequential and parallel implementations of our performance test programs to determine the best approach.

SYSTEM IMPLEMENTATION AND ARCHITECTURE

Implementation

In this last part, we are interested in the tools used for the realization of our application as well as the main interfaces of the application.

Choice of hardware and software

Hardware environment

In order to carry out our research project, we have used the following materials (Bopatriciat Boluma Mangata et al., 2021):

- ✓ Three laptops (LAPTOP) from the HP EliteBook brand.

Here are the characteristics of this machine:

- ❖ Mark : HP EliteBook ;
- ❖ Operating system: Windows 8.1 Professionnel 64 bits ;
- ❖ Processor : Intel (TM) Core i5 1,70 GHz, ~2,40 GHz ;
- ❖ RAM Memory capacity : 8 Go ;
- ❖ Hard disk capacity: 300 Go.

These computers contain a biometric application in C# that allows instructions to be given to the Arduino card via the serial port and a database replicated in three different instances representing our three sites.

Hardware architecture of the system

The material architecture of the project is as follows (Bopatriciat Boluma Mangata et al., 2021):

- ✓ Personal Digital, a fingerprint reader, communicating through the USB port ;
- ✓ A computer, containing a biometric application in C# that allows instructions to be given to the Arduino card via the serial port and a database replicated in three different instances representing our three sites.
- ✓ The Arduino card, which is programmed to analyse and generate electrical signals, in order to carry out automatic door opening and closing tasks (access control).
- ✓ TOWER PROTM Micro Servo 9g SG90, a stepper motor that will allow us to make the opening and closing movements of the doors.

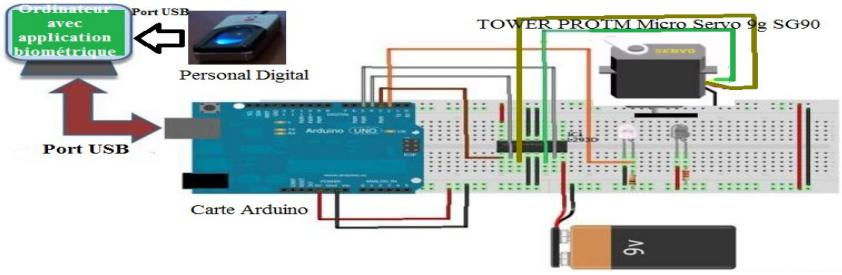


Figure 1: Hardware architecture of the system

RESULTS OBTAINED

Here is a representation of some of the graphical interfaces of our application:

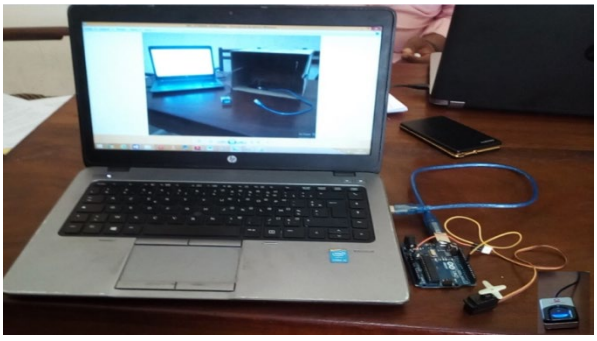


Figure 2: The material tools of our project

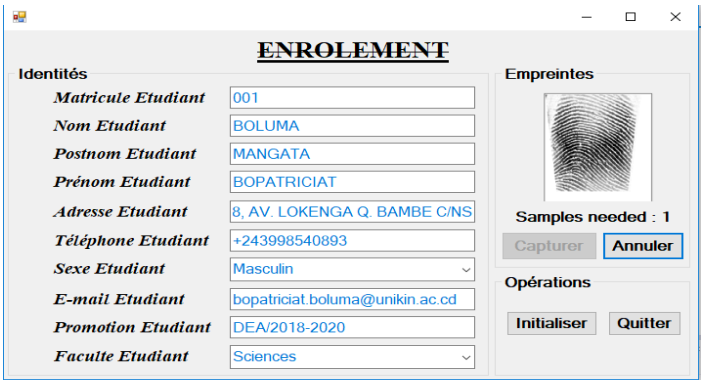


Figure 3 : The enrolment window

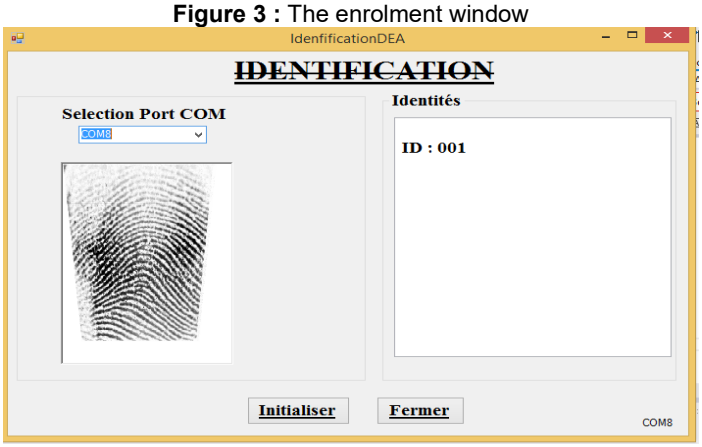


Figure 4: The identification window with a valid fingerprint

Interpretations of the results obtained

The table below represents the different values of execution time of the sequential version on the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600.

Table 1: Sequential version run times

Data size	Execution time(ns)
50	0,11
100	0,26
150	0,39
200	0,53
250	0,67
300	0,81
350	0,95
400	1,09
450	1,23
500	1,37
550	1,51
600	1,65

The following graph is intended to evaluate the execution times of each data size when responding to the sequential test program (Melnykov, Chen & Maitra, 2012)

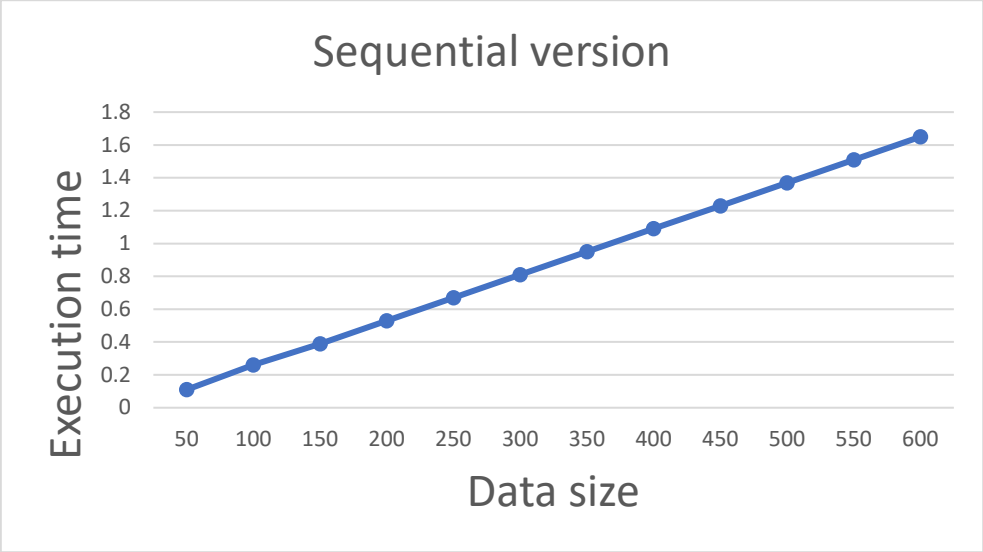


Figure 5: Execution time of the sequential version

The table below represents the different values of execution time of the parallel version on the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600.

Table 2: Parallel Version Execution Times

Data size	Execution time (mn:s:ms)
50	0,05
100	0,12
150	0,18
200	0,25
250	0,31
300	0,38
350	0,44
400	0,51
450	0,57
500	0,64
550	0,7
600	0,77

The following graph is intended to evaluate the execution times of each data size when responding to the parallel test program (Li, Peng, Su & Jiang, 2020).

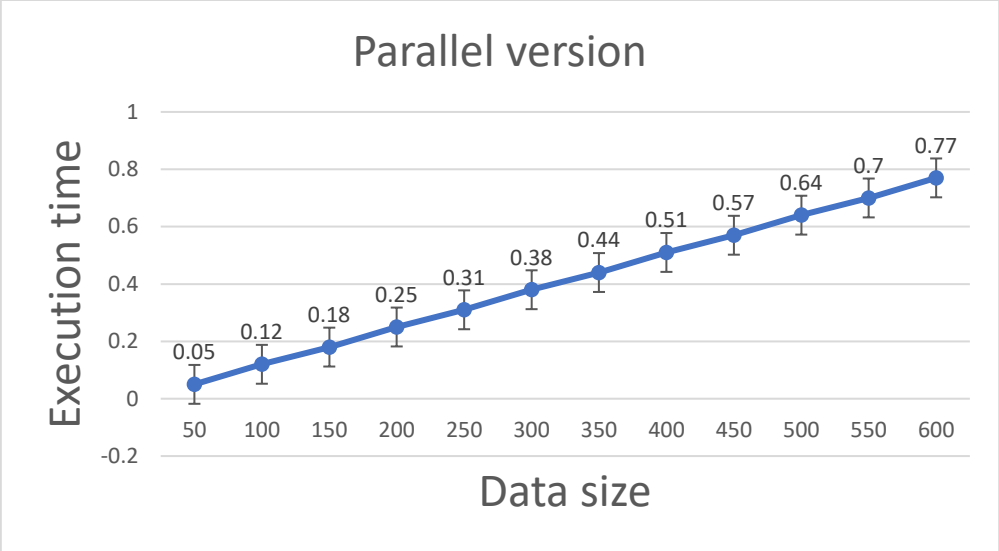


Figure 6: Execution time of the parallel version

The table below represents the different values of execution times of the sequential and parallel versions on the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600. The aim is to evaluate the execution times of each data size by responding to the sequential and parallel test program to determine the best approach (Wan & Zou, 2017).

Table 3: Sequential and parallel execution times

Data size	Sequential execution time (mn:s:ms)	Parallel execution time (mn:s:ms)
50	0,11	0,05
100	0,26	0,12
150	0,39	0,18
200	0,53	0,25
250	0,67	0,31
300	0,81	0,38
350	0,95	0,44
400	1,09	0,51
450	1,23	0,57
500	1,37	0,64
550	1,51	0,7
600	1,65	0,77

The graph below allows us to interpret the results obtained by the sequential and parallel implementations of our performance test programs to determine the best approach (Dall'Olio, Curti, Fonzi, Sala, Remondini, Castellani & Giampieri, 2021). It is very clear that the sequential program is too greedy in terms of computation time compared to the parallel program which minimizes the computation time (Rosenberg, Mininni, Reddy & Pouquet, 2020)

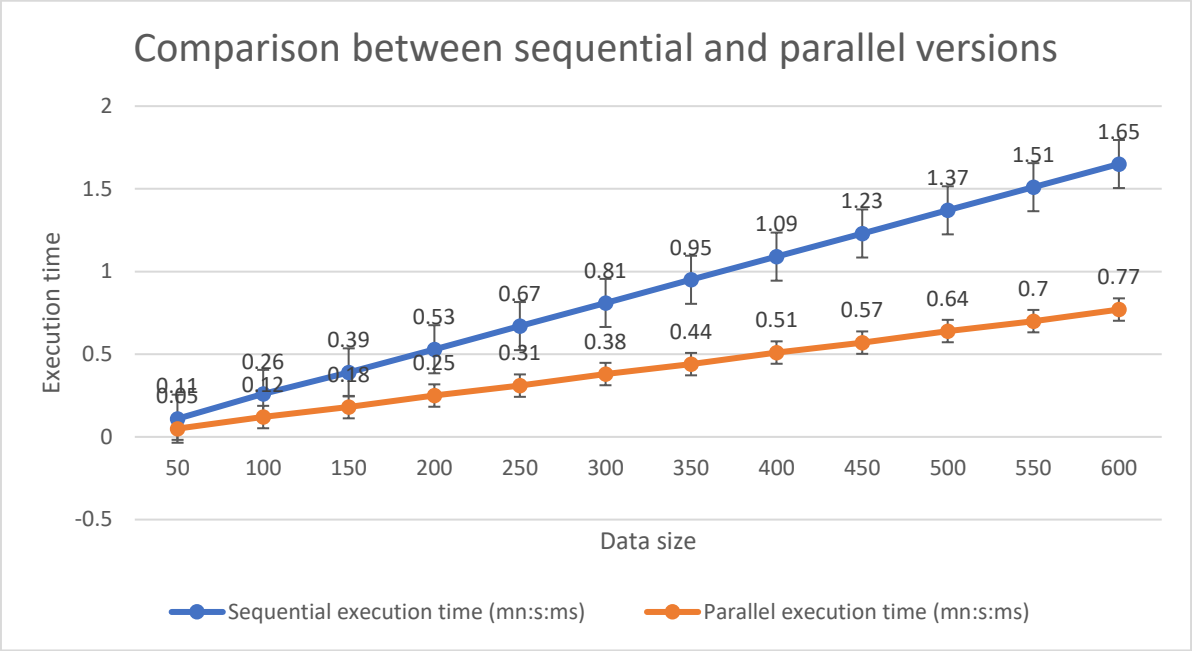


Figure 7: Comparison between sequential and parallel execution times

CONCLUSION

We have come to the end of our work which allows us to evaluate the execution time performance of a single mode biometric recognition system for access control to premises secured by fingerprints.

To speed up the computation time in this system, we resorted to parallel programming, targeting more loops in the verification module.

Our approach was therefore to parallelize all loops that are computationally intensive during the verification of fingerprints in the database.

For this, we exploited Microsoft's Task Parallel Library, specifically exploiting the loop for and loops for each.

On the test set performed in sequential and parallel versions in the different data sizes, namely 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, and 600, we can state that the results obtained by the sequential and parallel implementations of our performance test programs allowed us to determine the best approach. Therefore, it is very clear that the sequential program is too greedy in terms of computation time compared to the parallel program which minimizes the computation time.

ACKNOWLEDGMENTS

The authors appreciate the reviewers for their contributions towards improving the quality of this research.

CONFLICT OF INTEREST DISCLOSURE

All authors declare that they have no conflicts of interest to disclose.

REFERENCES

Abdellatif, M. (2016). Accéleration des traitements de la sécurité mobile avec le calcul parallèle (Doctoral dissertation, École de technologie supérieure).

Bopatriciat Boluma Mangata & al. (2022). Performance evaluation of a single access contol system. journal of research in engeneering and applied sciences. Volume (7 Issue 01), p4-6.

Bopatriciat Boluma Mangata et Al.(2021). Contribution of an Embedded and Biometric System in a Replicated Database for Access Control in a Multi-Entry Institution. International Journal of Science and Research (IJSR), Volume (10 Issue 3), p2-5.

Dall'Olio, D., Curti, N., Fonzi, E., Sala, C., Remondini, D., Castellani, G., & Giampieri, E. (2021). Impact of concurrency on the performance of a whole exome sequencing pipeline. BMC bioinformatics, 22(1), 1-15.

Fryza, T., Svobodova, J., Adamec, F., Marsalek, R., & Prokopec, J. (2012). Overview of parallel platforms for common high performance computing. Radioengineering, 21(1), 436-444.

Li, C., Peng, Y., Su, M., & Jiang, T. (2020). GPU Parallel Implementation for Real-Time Feature Extraction of Hyperspectral Images. Applied Sciences, 10(19), 6680.

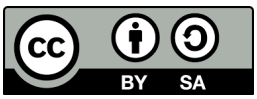
Melnykov, V., Chen, W. C., & Maitra, R. (2012). MixSim: An R package for simulating data to study performance of clustering algorithms. Journal of Statistical Software, 51, 1-25.

Miao, Y., Tian, Y., Peng, L., Hossain, M. S., & Muhammad, G. (2017). Research and implementation of ECG-based biological recognition parallelization. IEEE Access, 6, 4759-4766.

Ocaña, K., & de Oliveira, D. (2015). Parallel computing in genomic research: advances and applications. Advances and applications in bioinformatics and chemistry: AABC, 8, 23.

Reumont-Locke, F. (2015). Méthodes efficaces de parallélisation de l'analyse de traces noyau (Doctoral dissertation, École Polytechnique de Montréal).

Rosenberg, D., Mininni, P. D., Reddy, R., & Pouquet, A. (2020). GPU parallelization of a hybrid pseudospectral geophysical turbulence framework using CUDA. Atmosphere, 11(2), 178.



Tavara, S., Schliep, A., & Basu, D. (2021, September). Federated Learning of Oligonucleotide Drug Molecule Thermodynamics with Differentially Private ADMM-Based SVM. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 459-467). Springer, Cham.

Wan, S., & Zou, Q. (2017). HAlign-II: efficient ultra-large multiple sequence alignment and phylogenetic tree reconstruction with distributed and parallel computing. Algorithms for Molecular Biology, 12(1), 1-10.

Williams-Young, D. B., De Jong, W. A., Van Dam, H. J., & Yang, C. (2020). On the Efficient Evaluation of the Exchange Correlation Potential on Graphics Processing Unit Clusters. Frontiers in chemistry, 951.