CS395 INTERNSHIP PROJECT REPORT: SECURE AND PRIVATE USER-BASED KNN RECOMMENDER

Elif Cemre Durgut

Sabancı University

Faculty of Engineering and Natural Sciences

Computer Science & Engineering

0026493

Supervised by Albert Levi

EPFL – Scalable Computing Systems Laboratory

Supervised by Rafael Pires

13/06/2022-26/08/2022

Submitted on 02/10/2022

ABSTRACT

EPFL is a Swiss public university that is ranked 6th among the other universities in Europe according to QS. It includes laboratories from very different fields with leading of outstanding professors. Scalable Computing Systems is one of the laboratories within the university that mainly works on privacy and security in machine learning specifically federated and decentralized.

In this project, the goal is to develop a private user-based K-nn movie recommendation system using Intel Software Guarded Extensions. Then, it aims to determine the overhead of SGX by comparing the app with and without SGX by latency-throughput tests.

At the end of the internship, the project is successfully completed, and the test cases are written. It is found that the time per request increases as the total number of requests sent by the client at a time increases for the non-SGX system. The comparison between non-SGX and SGX systems continues.

TABLE OF CONTENTS

1. INTRODUCTION

Recommender systems play an important role in both customers' and commercial websites' life. However, those systems have the potential risk to leak user profiles which endangers users' privacy.

This project aims to develop a secure user-based K-nearest-neighbor recommendation system that recommends movies to users without leaking their private information. This privacy preservation is achieved using Intel Software Guarded Extensions (SGX).

Firstly, information about the company and its departments is given. Later, the project background is explained in detail in terms of the motivation behind the project, and related literature review from different sources.

In the internship project part, the project content, the boundaries, the technologies that have been used, the project flow, and my responsibilities will be discussed in detail. Then, I give more information about how this internship affected my future career plans, what I have learned, and the difficulties that I faced during my internship.

Finally, the conclusion is given which explains the major points of my internship and my recommendations for future PROJ302 students. The references can be found in the end. This part concludes the internship report.

2. COMPANY INFORMATION

Full title: Swiss Federal Institute of Technology Lausanne (Ecole Polytechnique Fédérale de Lausanne)

Address: CH1015 Ecublens, Lausanne, Switzerland

Contact Telephones: + 41 21 693 11 11

Web page: https://www.epfl.ch/en/ (University Web page)

https://summer.epfl.ch/ (Summer@EPFL Program Web page)

EPFL is a public research university in Lausanne, Switzerland which is controlled by the Swiss federal government. In 1853, a few students started at the Ecole Spéciale de Lausanne in subjects such as chemistry, physics, mathematics, drawing, architecture, and civil engineering. In 1890, the Academy was transformed into a university. In 1945, it changed its name to Ecole polytechnique de l'Université de Lausanne, in other words EPUL. The university Ecole Polytechnique Fédérale de Lausanne was officially founded in 1969.

As of 2022, the university has more than 17000 students and employees where the 30% of the students are female and 57% foreign nationals. There are around 500 laboratories and research groups in various disciplines such as math, life sciences, architecture, and engineering. It is ranked 14th in the QS World University ranking, alongside Columbia and Yale Universities.

The university offers five schools and two colleges:

School of Architecture, Civil and Environmental Engineering (ENAC): The faculty grasps major challenges such as climate change, global digitalization, and rapid

urbanization in its entirety whilst considering economic, technical, and social demands.

School of Computer and Communication Sciences: The research areas of the faculty include AI & ML, Human-Computer Interaction, Algorithms, Information & Communication Theory, Digital Education, Computer Architecture, Systems.

School of Basic Sciences: It provides an educational and research environment in mathematics, physics, chemistry, chemical engineering.

School of Engineering: It aims to lead in research discoveries in engineering; to support and contribute to technology transfer; to innovate in engineering education; and to contribute engineering solutions to societal challenges.

School of Life Sciences: It mainly studies biology to advance understanding the living world and solve biomedical problems.

College of Management of Technology: It focuses on research and teaching in the areas of Management Science with close ties to Engineering and Technology.

College of Humanities: It encourages research and educational programs that combine the human and social sciences with engineering, life sciences, natural sciences.

One of the main competitors of EPFL is ETH Zurich which is one of the prestigious universities in Switzerland that holds the 8th place in the QS World University rankings.

3. PROJECT BACKGROUND

3.1 Department Information

Two bachelor programs, one in Computer Science, one in Communications, and four master's programs in Comp. Sci., Communication, Data Science and Cyber Security are offered.

There are 55 professors, 267 Ph.D. students, 666 master's students, and 1036 bachelor's students currently studying/working at the IC department.

At the faculty School of Computer and Communication Sciences, there are 48 laboratories mainly divided into three Systems, AI, and Theory. The labs have a wide range of subjects from ML, and distributed systems to security, and computer graphics. The full list of laboratories can be found on this link.

The laboratory that I have been working with is SaCS in other words Scalable Computing Systems which is led by Prof. Anne-Marie Kermarrec. One professor, two post-docs, four Ph.D. students, and 8 trainees (including me) work in SaCS Lab. Some of their current projects are on decentralized federated learning, trusted executed environments, and differential privacy. They also have collaborated on projects with Distributed Computing Lab (DCL) which is led by Rachid Guerraoui.

Their research interests are system support for machine learning, federated learning systems, large-scale recommenders, graph-based systems, privacy-aware recommendation systems, and collaborative computing.

Anne-Marie Kermarrec

- SaCS Lab Professor
- Associate Dean for Education

  anne-marie.kermarrec@epfl.ch

Rafael Pires (My project supervisor)

- Post-doc at SaCS
- Scientist

  rafael.pires@epfl.ch

Diana Andreea Petrescu

- Ph.D. student at DCL who will continue to my project
- diana.petrescu@epfl.ch

3.2 Status of the Project

In addition to the full professor position, Anne-Marie is the CEO of Mediego app which is a start-up that mainly works on recommendation systems. With Mediego, you can send highly customized content to your customers, and identify their preferences, interests, and their potential to purchase, all in real-time.

Earlier this year, my supervisor and the others from the same lab published a paper about a TEE-based decentralized recommender system called REX which uses a trusted execution environment provided by Intel (SGX) to make effective recommendations while preserving privacy which I will give more information on literature review part.

3.3 Motivation and Project Description

3.3.1 Motivation

Recommendation using machine learning to recommend any type of items such as books, movies, news, Instagram posts, or advertisements to users is one of the hot topics nowadays. Especially, in today's world, there is new content to see, new movies to watch, and new clothes to buy almost every second. With such voluminous and fast-growing data, it is hard for users to decide and pick. That's why recommendation system plays an important role for both users and companies. In addition to helping the users to choose the right fit among many options, they also help companies to increase their profits by recommending the right product at the right time to the right user.

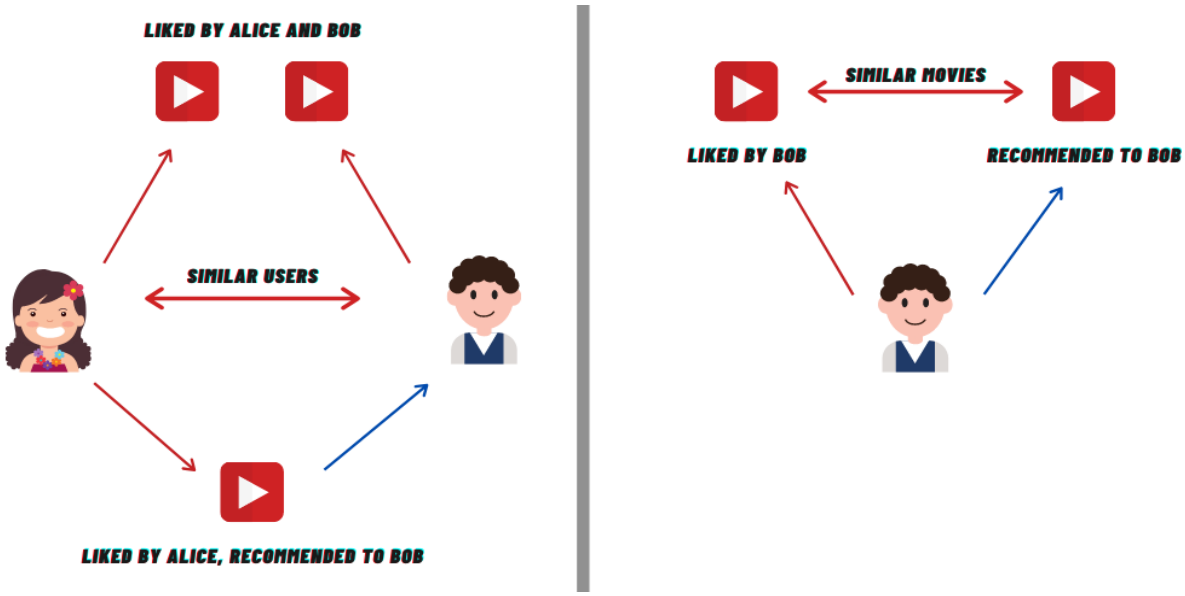

Figure 1: The comparison between collaborative and content-based filtering (Anand, 2020)

There are mainly two ways for recommendation systems, one is item-based and the other is user-based. In item-based recommending, the recommendations are based on the item similarities. As it is shown in the right side of Figure 1 above, if the user likes a movie, other similar movies are recommended to that user. One the other hand, the other way to recommend items to users is user-based recommendation also known as collaborative filtering in which the similarities between user profiles are used to learn user preferences and make predictions for other users as shown in Figure 1 on the left. However, this approach poses a serious threat since the recommender system learns users' preferences. One of the ways to solve this is differential privacy by adding noise to the data. However, this approach creates a dilemma between accuracy and privacy.

Therefore, increasing the accuracy without losing privacy is an important thing to achieve in recommenders.

3.3.2 Project Description

This project aims to develop a software application that uses SGX trusted enclaves provided by Intel to share data. SGX (Software Guard Extensions) helps to protect the data and to keep and operate on it in an isolated private region of the memory which is called an enclave. The goal of the project is to recommend movies to users by comparing the user profile to other profiles to find similarities and more importantly doing this recommendation in a secure way.

3.4 Related Literature

Kermarrec et. al. (2022, p.1) explains the importance of recommender systems in today's world in their article TEE-based decentralized recommender systems: The raw data sharing redemption. Lately, the data has been growing exponentially and this situation puts the consumers/users in a difficult position in terms of considering all the options available and making the best choice.

As they mentioned (2022, p.2), personalized recommendation systems are mainly collaborative filtering, matrix factorization, and deep neural networks. Their focus in this paper is collaborative filtering, a quite popular algorithm lately, which works on the matrix A_{mxn} that represents m users and n movies where the entries are movie ratings. And the goal is to fill up the matrix with rating predictions to make recommendations later. However, CF exploits the similarities between user profiles which is a serious privacy threat. Therefore, CF faces the issue of sacrificing the accuracy and keeping the privacy. This is where the Intel SGX takes an important role by preserving users' privacy.

They worked on a movie recommendation system using the MovieLens dataset on Intel SGX. However, their project differs from mine because theirs focuses on decentralized machine learning and the difference between model sharing and raw data sharing with SGX.

In their papers, they found that raw data sharing with SGX is more efficient and has higher accuracy & privacy compared to model sharing. However, they assume that the most-Intel based laptops have SGX which allows private communication and data sharing. But Intel has announced that SGX has been removed from Intel 11th generation and newer CPUs. Therefore, it cannot be assumed that the users in this decentralized SGX system (REX) have SGX on their CPU's.

The same privacy-accuracy tradeoff problem is approached by Shokri et. al. in their paper Preserving Privacy in Collaborative Filtering through Distributed Aggregation of Offline Profiles from 2009. They propose a method to avoid privacy violation where users have both online shared with the server and offline hidden from the server profiles. The users update their online profiles frequently in an independent and distributed way. Then, each user contacts another user arbitrarily and updates her offline profile in an aggregated manner.

I also searched for machine learning examples with SGX and found some applications that I can get help to build my own app. One of them is the code by a previous student at EPFL who works with SaCS on his paper Privacy-preserving model aggregation in decentralized machine learning with hardware enclaves where he aims to preserve the privacy while sharing models in decentralized learning.

4. INTERNSHIP PROJECT

4.1 Project Objective

As mentioned in the project motivation part, one of the most effective recommender system schemes is collaborative filtering (CF) which analyzes similarities between user profiles and makes recommendations based on those similarities. However, CF recommenders expose users' private data because they involve access to sensitive data through user profiles. Such profiles are composed of user ratings on items and may therefore leak users' personal features or preferences. To address the problem, this project aims at improving privacy in KNN-based recommenders at the inference step. This application should make sure that no personal information is leaked. This will be achieved with the aid of Intel Software

Guarded Extensions (SGX), a trusted executed environment that provides application enclaves with automatic memory encryption and other security features. In addition, we will investigate techniques such as differential privacy against statistical attacks on aggregated data and Byzantine fault tolerance against malicious legitimate users. This last part will be done in collaboration with Distributed Computing Lab.

4.2 My Responsibilities

My first responsibilities were to make research about private Knn models, differential privacy, and SGX in the first weeks to have a general idea. Later on, it was to build a user-based Knn model by comparing user profiles and making suggestions to users with MovieLens dataset.

My next responsibility was to upload my Knn graph to the enclave at the initialization. This graph shows the most similar k users to each user. Then, to create server in C++ and clients in Python and make the communication between them using ZMQ. The privacy of user data is secured with the help of Intel SGX in trusted enclaves. Finally, to measure the latency-throughput statistics for the servers with and without SGX and compare these graphs to find out the overhead of adding SGX.

4.3 Methodology

To build a user based KNN recommender, the MovieLens dataset will be used. (See the ml-latest-small dataset in this link.) The KNN graph is then supposed to be loaded in an SGX enclave for the inference step. This enclave will receive requests for recommendations from a user and will query the users' K nearest neighbors for their profiles, before providing the recommendation. The communication between the server and the users will be achieved using ZeroMQ.

EPFL machines with SGX capability will be used for implementing hardware enclaves. The development will be made using C, C++, and Python languages on Ubuntu.

4.4 Expected Outcome

The outcome of the project is planned to be a movie recommender system using Intel SGX. The system with and without SGX will be compared and analyzed to see if using SGX is worth or not. This will be achieved through latency-throughput tests. The test will be done in steps, firstly 10 clients will send requests at a second, then 100, and finally 1000 requests to observe if the server is able to handle many requests.

4.5 Details

First of all, I would like to start by explaining the technologies that I used in my project which are Intel SGX and ZeroMQ.

1) Intel SGX

In this section, I will briefly explain what Intel SGX is, how it works, some terminology, and later how I used it in my project.

SGX (Software Guard Extensions) is a technology provided by Intel that protects data via a unique application isolation system. The data to be protected is modified only inside of enclaves. The enclave is an encrypted region, and it is decrypted inside the processor. SGX protects the enclave code and data from OS, hypervisor, BIOS, drivers, and any remote attack.

Terminology

Trusted: Code that runs inside an enclave.

Untrusted: Code that runs in the application environment outside of the enclave. The operating system(OS) and the virtual machine manager(VMM) are considered untrusted in the SGX case.

Ecall: Refers to enclave call. It is a function call from the application that enters the enclave.

Ocall: Refers to the outside call. It is a function call made from within the enclave to outside (untrusted part).

A summary of the flow and steps can be found below in the figure:

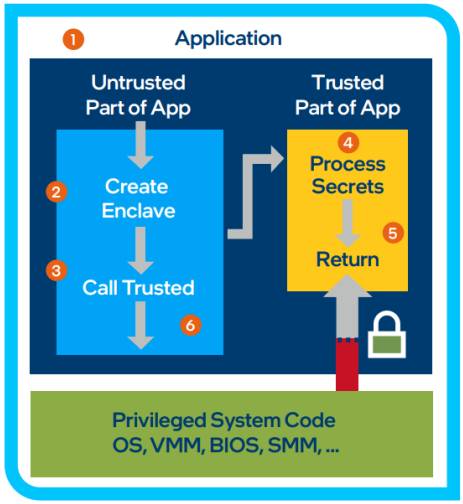

Figure 2: Intel SGX Working Scheme (Intel, 2016)

1) The application is built with two parts, an untrusted part and a trusted one.
2) The untrusted part of the app creates the enclave.
3) Again, the untrusted part makes an ecall and the execution is moved to the trusted part of the app.
4) Enclave runs and processes the secret data; no external access is allowed.
5) The trusted function returns the result via ocall and the enclave data remains protected in the enclave.
6) The rest of the execution continues in the untrusted part.

As stated in the figure above, the enclave is created in the application. While the creation of the enclave is done by using the sgx_create_enclave function, the sgx_destroy_enclave function is used to destroy the enclave in the end.

To compile the enclave, a simple enclave.edl file is needed to define trusted and untrusted functions. The content of the enclave.edl file is as follows:

```
enclave{
     trusted{
          //trusted functions
     };
     untrusted{
          //untrusted functions
     };
};
```

Figure 3: Generic Enclave.edl File

EDL defines several attributes that can be used with pointers. The most commons are in, out, string and size.

The [in] and [out] serves as direction pointers. [in] pointer is used when the parameter is passed from the calling procedure to the called procedure. For an ECALL, the parameter with in pointer is passed from the application to the enclave. For an OCALL, it is passed from the enclave to the application. And the reverse applies to the [out] pointer. [in, out] combines [in] and [out] meaning that the data is copied back and forth.

After defining the functions in the edl file, the Edger8r tool creates some files for compilation. Edger8r parses the EDL file and generates the trusted and untrusted bridges and proxies to interface between the application and the enclave. As can be

seen in the figure below, it creates four files namely, enclave_u.h, enclave_u.c for the untrusted part, and enclave_t.h, enclave_t.c for the trusted part. The figure below shows the trusted and untrusted parts of the application.

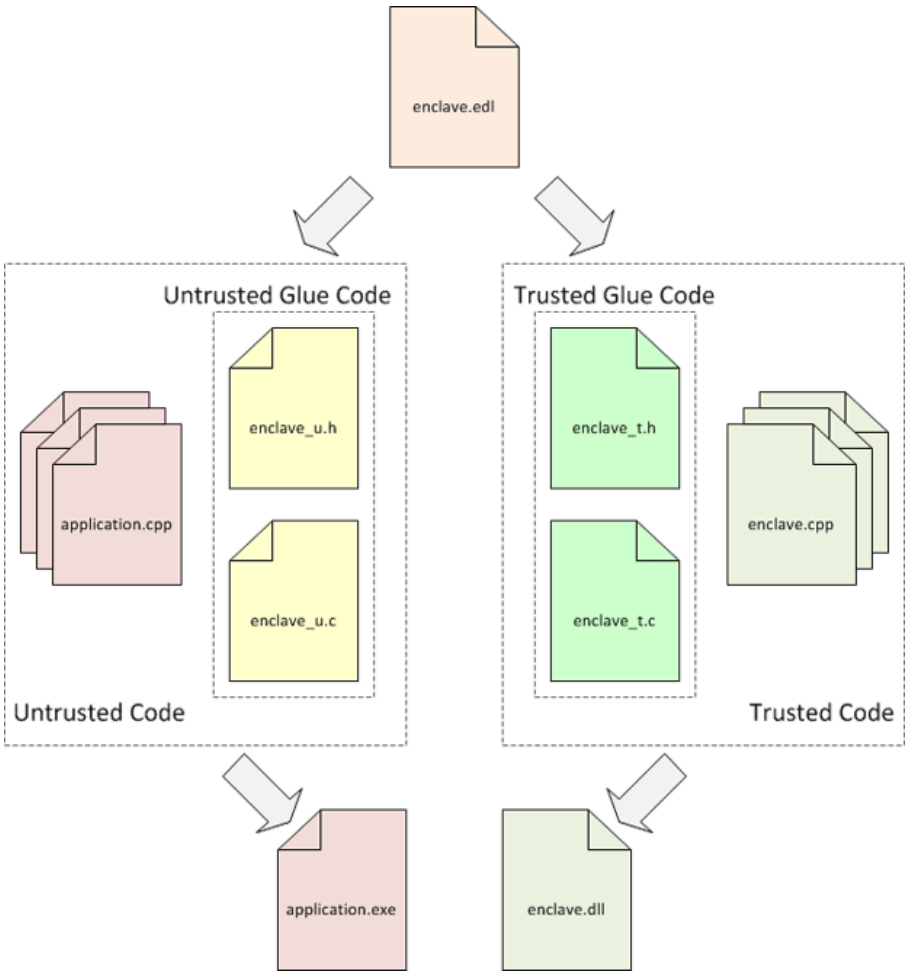

Figure 4: Edger8r Tool Scheme (Intel, 2016)

2) ZeroMQ (ZMQ or ØMQ)

ZMQ is an asynchronous messaging library that allows communication between applications even if they are written in different programming languages. It is known for being fast, lightweight, and its high performance.

As they stated on their website, the zero in ZeroMQ stands for zero message broker, zero latency, zero cost, and zero administration.

It supports different types of messaging patterns such as Request-Reply, Publisher-Subscriber, Dealer-Router, and Push-Pull.

My project flow

In this part, I would like to mention how I arranged my project steps, some of the problems that I encountered, and how I approached and made some design decisions.

I started the project by researching trusted execution environments, differential privacy, and decentralized machine learning. A week later, my supervisor and I decided on the boundaries of the project.

I built two K-nn models in Python, an item-based and a user-based model. Item-based K–nn model compares movies and suggests movies by comparing the similarities which are calculated by user ratings. User-based K-nn model compares the user profiles by analyzing the movie ratings. Then, it gets the top k most similar users and extracts the movies they have rated and makes recommendations based on that as can be seen in Figure 7. I extracted the K-nn graph as a txt file which includes the user ID and corresponding similar user profiles ID for them. I used the MovieLens data for movies and ratings.

Later, I ran and analyzed a HelloWorld SGX example to understand how to do ecalls and ocalls, also how to load data to the enclave. I loaded my K-nn graph that stores the neighbors for users to the enclave since it should be kept protected. This loading is done during the initialization of the enclave in the ecall_init function that you can see in Figure 5, 6, 7 below. The declaration of this function is in Figure 6, the definition is in Figure 7 and it is called in Figure 5.

Pseudocode of my application

You can reach my full project via this link on GitLab.

```
define ocall_receive()
define ocall_send()
main
      initialize enclave
      the open knn graph file
      ecall_init()
      destroy enclave
```

Figure 5: App.cpp File

```
trusted
{
      ecall_init([in, string] char* knn_graph)
}
untrusted
{
      ocall_print([in, string] char* str)
      ocall_receive([out, size] char* message, size_t   size)
      ocall_send([in, string] char* message)
}
```

Figure 6: Enclave.edl File

```
ecall_init() :
      save the Knn graph
      while True:
            receive a message
            if the message is movie request:
                  ask other users to send their profiles
            else if the message is hello
                  send "OK"
            else if the message is incoming movies:
                  save the movies
                  calculate average movie ratings by id
                  if all the neighbors are communicated:
                        get the top 5 movies and send
                  delete the used user profiles
            else:
                  invalid request
```

Figure 7: Enclave.cpp File

Then, I started to research ZeroMQ and again did a hello world example with the REQ-REP pattern in which the server (REP socket) is in C++ and the client (REQ socket) is in Python. However, the classic REQ-REP pattern does not meet all the requirements for my project such as sending not only replies but also requests to clients by the server and sending those requests to specific clients only by using their IDs. Also, the REQ-REP pattern is synchronous meaning that it forces you to reply to the clients in order.

To meet my requirements, I searched for other ZMQ patterns like PUB-SUB or PUSH-PULL. In the PUB-SUB pattern, the publisher sends each message to "all of many" clients. However, I should not send my messages to all clients. On the other hand, in PUSH-PULL pattern distributes messages to its pull clients evenly, in this case, consumers (clients) are not differentiable, unlike my clients.

Finally, I found out about the Dealer-Router pattern which is a powerful version of REQ-REP where it allows the server to talk to multiple clients asynchronously. I set IDs for each client so that the server can communicate with clients via their unique IDs.

I decomposed my project into small parts as I learned and taught in IF 100.

Therefore, I decided to make the project in the following order:

1. without SGX with the server and 1 client
2. without SGX with the server and multiple clients
3. with SGX with the server and multiple clients

First of all, I wrote my ZMQ class in C++ to use in my project. I designed the ZMQ constructor such that it is possible to declare my ZMQ socket object globally and use it in different functions including the main one.

I self-assigned IDs to clients as the same as their IDs in MovieLens dataset so that I do not need to store a map that keeps client ids and user ids.

Then, I managed to make the first step work. However, I had problems with running multiple clients at the same time. Because, even for one recommendation request, I need k+1 (k: k nearest neighbor, 1: the requester client) for many clients. I used to open many terminals and connect a client per terminal. But this method is not feasible when I need to run all the clients. Therefore, I searched for a way to run programs in parallel and I found GNU Parallel. In this way, I am able to run multiple clients at the same time.

In the final part, I did some latency-throughput tests. I created a simulation where the clients send 10, 100, and 1000 requests at a time to see how my project handles these many requests. I tested these scenarios in both my applications with SGX and without SGX.

4.6 Results

At the end of my internship, I completed my Knn-recommender both with SGX and without SGX versions. However, I could not find the time to do the respective tests. After my internship has officially ended, I continued to work on my project and finished the test cases. I ran the tests for the non-SGX recommender, and the figure below shows the time vs number of requests graph. In the test cases, I created a simulation where 100 users are active and 10 of them send recommendations 10, 100, and 1000 requests at a time.

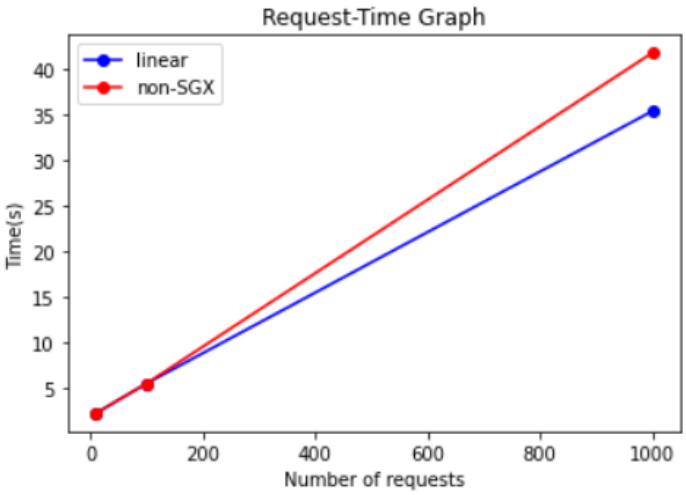

Figure 8: Number of requests vs Time(s) graph in points 10, 100 and 1000 on the x-axis

The red line shows the time taken for the server to reply to all the recommendation requests by clients. For comparison, I drew a linear line as well which is in colored blue. As it can be seen in the figure, as the number of requests per time increases, the time required to complete one request on average increases.

I completed the test cases for the SGX recommender as well. However, I cannot run it on my laptop. Therefore, I sent my project to my supervisor for him to run it on EPFL IC Cluster machines and send the results to me.

My laboratory (SaCS) and DCL Labs will be continuing this project by adding differential privacy and Byzantine Fault Tolerance.

5. INTERNSHIP EXPERIENCE

5.1 Learning

My internship has gained me both hard skills and soft skills. I have read more articles and learned more about current research topics in different areas of machine learning. I have learned how to use two main technologies Intel SGX and ZeroMQ and gotten better at Linux and Git commands. On the other hand, as a soft skill, I have gotten better at project management, communication in a lab environment, being more planned, and being able to put realistic deadlines for myself and being committed to them.

My internship slightly changed my future career plans. It helped me to see how Ph.D. student works, their duties, projects, and exams. I have learned that after the first year, they decide what topics to search on, and what projects to do. I was considering applying for Ph.D. programs, but I do not think that I am experienced enough to do this right after my bachelor's. That's why I decided to do a master's first and think about Ph.D. later.

5.2 Relation to undergraduate education

SGX applications are written in C++, therefore knowing C++ and object-oriented programming from CS 201 and CS 204 courses was an advantage for me. Also, I was already familiar with the K-nearest neighbor algorithm from my CS 210 and CS 412 courses.

Also, my course project in CS 408 helped me to understand better the communication between the server and the users by using ZMQ. Because I already knew how to receive messages and how to handle the requests.

5.3 Difficulties

I have not used Intel SGX before. Since it does not allow any I/O operations such as even printing to the console, it was hard for me to see the results and debug.

Also, I was not so familiar with Linux commands except for basic ones. Therefore, I had some difficulties from time to time while running the code and writing the Makefiles or script shells.

5.4 A typical day

My day at work starts around 8.30-9.00 am. I work in an office that I share with other interns from my lab and two other labs. Firstly, I usually revise what I have done on the previous day and then I make a plan as an outline to see how much I need to progress to achieve my weekly goals. I use my laptop to work and connect to one of the computers at the IC cluster via ssh.

On Mondays, we have a lab reading group session in which one of the members of SaCS presents a scientific paper for us to know about what's happening in CS around the world. Because it is not possible to read all the papers published considering that thousands of papers about ML and/or privacy are published each year. These sessions are organized every two weeks with the leadership of Akash Dhasade, one of the Ph.D. students at SaCS.

On Tuesdays, we have a weekly laboratory meeting, which everyone joins, at 10 am. In these meetings, we discuss what everyone has been doing since the last meeting, their problems, and any interesting ideas/findings that they might want to share. Those meetings enable us to know about each other's projects and progress. Also, we sometimes use these meetings for master students' final project

presentations or Ph.D. students' candidacy exam preparations. Those meetings usually last for an hour.

Again on Tuesdays, we have a weekly meeting to discuss the project that I am involved in with two Ph.D. students and a post-doc from Distributed Computing Lab (DCL), my supervisor, and the professor from SaCS at 11 am after the lab meeting. However, the Ph.D. student who will contribute to the project had to finish another project first. Therefore, these meetings lasted for three weeks only. After that, we had weekly meetings only with my supervisor to discuss my progress, the problems/bugs that I encountered, and my solution ideas.

After the meetings, I work for one and half an hour more then we have a lunch break at 12.30 which lasts about an hour where we eat together with people from my lab in the cafeteria. After lunch, I continue to work until 17.00-18.00 depending on my workload that day. I go to my supervisor's office if I cannot solve an error or if I have some questions about an implementation issue.

Before I leave the office, I keep notes about what I have done that day and the current problem so that I can start from there directly the next day.

6. CONCLUSIONS

Recommender systems play an important role in terms of increasing the profit of commercial websites and helping customers to pick what they want to buy, watch, or eat.

User-based recommenders are one of the efficient ways to make recommendations for users by comparing their profiles to others and suggesting the

items that their neighbors liked. However, this creates a privacy threat since recommenders reach and store user profiles. To overcome this problem, I developed a user-based Knn recommender with Intel SGX that keeps user profiles in trusted enclaves securely.

In the end, I tested my recommender systems with both SGX and without SGX where 100 clients are active and sending 10, 100, and 1000 requests at a time respectively. I found out that the time taken per request increases as the total number of requests increases for the application without SGX. The comparison between SGX and non-SGX continues. I expect the application with SGX take more time compared to non-SGX however I assume this overhead will be negligible when it's considered that it provides a private and secure environment.

7. RECOMMENDATIONS

I am so glad that I joined Summer@EPFL Program. It helped me to see the academic research environment, make very good connections with post-docs and Ph.D. students, and travel to Switzerland without economic concerns. The applications close in late November, therefore I suggest students pay attention to this date.

My friends who applied for professors or labs individually were complaining about the lack of socializing. Therefore, I suggest students who are interested in Erasmus+ internship join such collective programs so that they can meet other interns from different countries and backgrounds.

My lab admin was so interested in all the administrative stuff and documents regarding visas. Also, my supervisor was always so helpful and ready to answer my questions.

Students who will conduct their internship at a university like mine should be aware that it is the summer period and most of the Ph.D. students and post-docs go on vacation for a month or so. Students should be prepared for working alone and finding solutions on their own.

Overall, I was happy to do my internship as a part of Summer@EPFL and I recommend it to everyone who wants to experience a good academic environment.

8. REFERENCES

Anand A. (2020). User-user collaborative filtering for jokes recommendation. Retrieved on 02/10/2022 from https://towardsdatascience.com/user-user-collaborative-filtering-for-jokes- recommendation-b6b1e4ec8642

Dhasade, A., Dresevic, N., Kermarrec, A., Pires, R. (2022). TEE-based decentralized recommender systems: The raw data sharing redemption.

Intel. (2016). Introduction to creating a sample enclave using Intel® Software Guard Extensions. Retrieved on 22/09/2020 from https://www.intel.com/content/www/us/en/developer/articles/code-sample/intel-software-guard-extensions-developing-a-sample-enclave-application.html

Intel. (2016). Intel software guard extensions developer guide. Retrieved on 20/09/2022 from https://download.01.org/intel-sgx/linux-1.7/docs/Intel_SGX_Developer_Guide.pdf

Pieter Hintjens. (2012). ZeroMQ guide. Retrieved on 20/09/2022 from https://zguide.zeromq.org/

Shokri, Reza & Pedarsani, Pedram & Theodorakopoulos, Georgios & Hubaux, Jean-Pierre. (2009). Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. 10.1145/1639714.1639741.