Automated Currency Exchange Machine Parallelization

Using Message Passing Interface

Heinner Amaro, Javier Pena, William Pennebaker

School of Computing and Information Science, Florida International University (FIU)

***Abstract:*** Nowadays, currency exchange processes around the world are an annoyance to the many tourists that try to use this service. Long queues at airports, lack of multiple currency exchanges and slow customer service all influence this process which is essential to a plethora of travelers around the world. With the advance of technology, it is worrisome that in 2019 airports only offer to its clients rudimentary Currency Exchange Kiosks when in this digitalized era, a simple ATM like machine would solve the issue. The ACEM is a fast, multiple-user machine that would make the currency exchange process, a faster and safer service to clients.

1. **INTRODUCTION:**

With the increasing rate of technology in our times, it is imperative to modernize as many processes as possible. We looked at the Currency Exchange issue as a golden opportunity to go ahead with our idea of using Message Passing Interface to modernize an already outdated processes that would be really beneficial to society. As of now, besides the Currency Exchange Kiosks that are offered by the airport and the local ATM machines that take only one customer at a time, there are not that many options for clients to have a safe and automated path to getting their currency exchange process done. We will implement the Automated Currency Exchange Machine to solve this issue. The idea of this machine will be similar to the one of an ATM, the catch here is that our machine will have the option to get an input of x= (2…N) where N is the number of currencies that the user would like to obtain back. Our first prototype of the ACEM is going to have a total of 6 different currencies. We are offering this flexibility to the customers in the case that the traveler has found himself using the airport as a stop to go to another

country and he need currency for the country they are staying on and the country they will be visiting in the future. Here’s where MPI comes into play, our machine is going to have a main CPU with multiple cores which are going to have different currencies assigned to them; the user will get a message to divide their input into different currencies so they can their currency back i.e. 100/2 currencies = 50. Our study will be carried out at Miami International Airport, we will take data on how many visitors does the airport get daily and then use that information to calculate the timing taken by customer when exchanging their currency to N=6.Depending on the speed of that process we would then identify how many ACEM’s would be needed to satisfy a normal day at Miami International Airport.

1. **RELATED WORK:**

In the field of ATM’s and currency exchange there hasn’t been that much related work, meaning that our project is both innovative and novice in this field. When it comes to ATM’s at airports and local towns there has been multiple research methods related to the security of it when it comes to fraud. Some research papers like a biometric (fingerprint) strategy to enhance ATM security in ATMs in India. At the time of transaction fingerprint image is acquired at the ATM terminal using high resolution fingerprint scanner.[1] There’s also been other studies to improve ATM security like face recognition techniques which recognizes your face from three different angles before accepting your transaction [2]. Lastly, there’s been some other studies related to having portable ATM machines that would make the transactions via satellite control to the bank, these would be battery powered. It allows its user to securely access bank and bank accounts and to securely effect either inter-account transfers or bill payments. Security is assured by assigning a PIN to each individual user and restoring it therein, by assigning a machine identity number to each machine and storing it therein, and by providing encrypted data communication. It has pre-stored there in the individuals account numbers and the phone numbers and other parameters needed for communication to the corresponding host computer bank. [3] As it may have been noticed, our project is completely different, we want to innovate and improve the world of ATM machines with the ACEM. Instead of the other previous mentioned studies that used biometrics, we would be using parallelization to get implement our idea. Our project focuses solely in the customer service and leaves aside the security issues. We are focused on speed and comfort, we want the users to get their money as fast as possible without having to worry about long queues and slow processes that would give them one currency at a time.

1. **PROCESS**

At the beginning of our brainstorming sessions, we were planning on having a machine with different inputs so that it would take N number of customers. This main master(process) of customers would then be divided into different slaves(mini-tasks) that would carry out the function of doing all the currency exchanges at the same time with different inputs. After brainstorming for a while, we thought this idea was not going to make it since it was not going to achieve the desired speed/results we wanted to obtain by doing this project. That’s when we switched to having different slaves carrying out different currencies. A normal user would come and input their money to then select which percentage they would like in n = 6 number of currencies. For this particular example, we would have the customer selecting two currencies (Yen, Pesos) and an amount of $200 dollars. The user then would make a partition of 100/100 to get 100 yen and 100 pesos respectively. Then we are going to have an array of amounts, each amount is going to be converted as follow. The array is going to be from amounts (5,10,15,20…n + 5), each currency is going to be converted in those amounts and then the ones specified by the user will be returned.

**void initializeArr(float\* currency) {**

**int i;**

**for (i = 0; i < N; i++) {**

**currency[i] = (i+1)\*5;**

**}**

We used a *converter(data,nmbr)* method that will convert the amounts on the array to each currency that we have on the system, the data parameter then will take the amount that will be partitioned into the other different amounts.

**void convert(double dollar, int cpucount) {**

**double peso, euro, yen, cad, lira, gbp;**

**printf("Dollar amount: $%.2f\n", dollar);**

**dollar = dollar \* .20;**

**peso = dollar \* 19.495;**

**printf("$%.2f = %.2f mex peso\n", dollar, peso);**

**euro = dollar \* 0.9028;**

**printf("$%.2f = %.2f euro\n", dollar, euro);**

**gbp = dollar \* 0.7736;**

**printf("$%.2f = %.2f gbp\n", dollar, gbp);**

**yen = dollar \* 108.3845;**

**printf("$%.2f = %.2f yen\n", dollar, yen);**

**cad = dollar \* 1.33191;**

**printf("$%.2f = %.2f cad\n", dollar, cad);**

**lira = dollar \* 5.7019;**

**printf("$%.2f = %.2f lira\n", dollar, lira);**

**}**

If the amount was $300 dollars then there will be partitions of $50 for each currency that will then be exchanged to another currency by using the 5,10,15…n+5 system. This all would be done using a MPI system, MPI or message process interface has as master and slaves, the master would take the process and send it to the slaves who would then execute the conversions along with the array all on a parallel way.

**void mpi(float\* currency) {**

**int i, othercpus;**

**MPI\_Status status;**

**int partitions = N / numcpus;**

**if (cpu == 0) {**

**for (othercpus = 1; othercpus < numcpus; othercpus++) {**

**MPI\_Send(&currency[partitions\*othercpus], partitions, MPI\_FLOAT, othercpus, 1, MPI\_COMM\_WORLD);**

**}**

**for (i = 0; i < partitions; i++) {**

**convert(currency[i], numcpus);**

**wait();**

**}**

**for (othercpus = 1; othercpus < numcpus; othercpus++) {**

**MPI\_Recv(&currency[partitions\*othercpus], partitions, MPI\_FLOAT, othercpus, 2, MPI\_COMM\_WORLD, &status);**

**}**

**}**

**else {**

**float data[partitions];**

**MPI\_Recv(&data[0], partitions, MPI\_FLOAT, 0, 1, MPI\_COMM\_WORLD, &status);**

**for (i = 0; i < partitions; i++) {**

**convert(data[i], numcpus);**

**wait();**

**}**

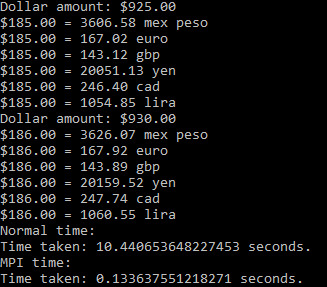
**MPI\_Send(&data[0], partitions, MPI\_FLOAT, 0, 2, MPI\_COMM\_WORLD);**

**}**

**}**

1. **RESULTS**

In order to test our results, we did not want just to test the speed of our problem, but at the same time we wanted to take our issue to a real-life study. Our test case in this occasion was Miami International Airport. Miami International Airport takes a 45,044,312 number of passengers yearly. That makes an average of 123409 daily. [4] As we can see that’s a really serious traffic of passengers in the airport on a daily basis. If we wanted to implement our machines in the airport we needed to carry out some calculations to find out how many hours of transactions could our produce in comparison to a sequential machine, so that way we could implement some other machines around the airport to satisfy the great demand of transactions that we would encounter. The running time for a $2000-dollar transaction divided by 6 currencies was 10.44 seconds as a normal procedure, while the one parallelized with a MPI took 0.13 seconds to finish.



In order to find out how may machines we would need around Miami’s airport we would need to multiply 123409 passengers by the number of seconds which is 0.13. That would give us a total of 16403 seconds/267 minutes/4.4 hours. On the other hand, the machine without parallelization would have taken 367 hours of transaction time. If we wanted to reduce the amount of transaction time in our airport, two machines would need to be implemented. That way 2.2 hour of transactions could be done

1. **CONCLUSIONS**

The MPI parallel functionality of Master and Slave was primordial in the timing success of our machine. Furthermore, our configuration steps of using an array to divide the different categories of conversions also contributed to the time complexity of our software. Overall, parallelism has shown a significant improvement in this currency exchange software. Looking into the future, we plan on using GPU multithreads system to try and improve the timings even more

1. **REFERENCES**

[1] <https://pdfs.semanticscholar.org/0d9f/f1280accdb90ee959fe02efe527b72ec375e.pdf>

[2] <https://ieeexplore.ieee.org/abstract/document/5942118>

[3] <https://patents.google.com/patent/US5050207A/en>

[4] <http://www.miami-airport.com/library/pdfdoc/Monthly%20Traffic%20Reports/December%202018%20Moving%20Twelve%20Months%20Traffic%20Report.pdf>