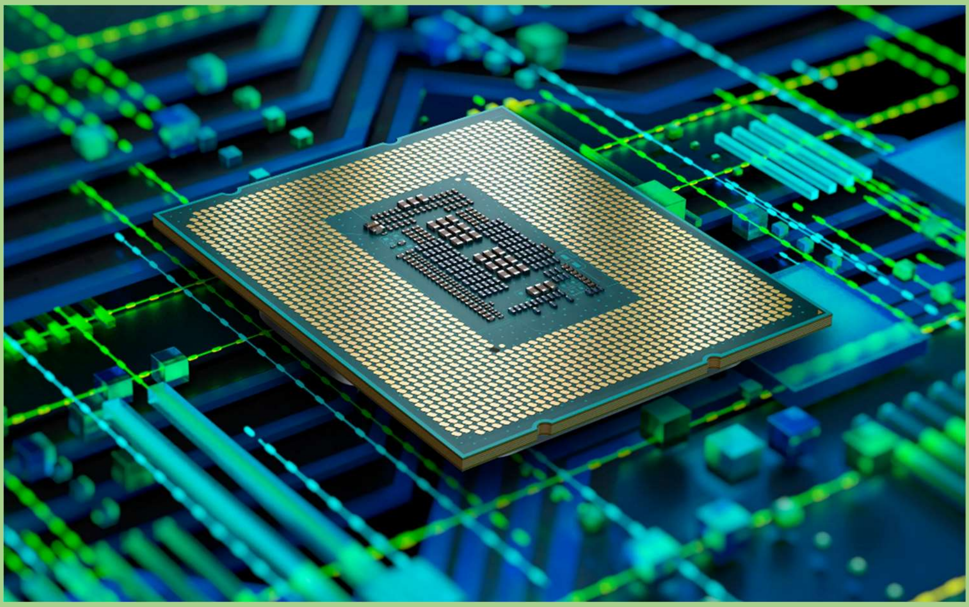
Measurement of the execution time of processes in different programming languages



Stan Andreea-Alexandra

Technical University of Cluj Napoca

October 24, 2022

Contents

1. Introduction

1.1 Context

The goal of this project is to measure the execution time of different processes in three distinct programming languages. The tested operations are: memory allocation, memory access (both static and dynamic), thread creation, thread context switch and thread migration. The obtained results from the measurements will be then compared between the programming languages.

This application can be used by people in order to decide which programming language would be the fastest, considering the operations mostly used by the program they want to create.

1.2 Specification

This application will be simulated in three different IDEs, one for each programming language chosen: Visual Studio 2019 will be used for C++, IntelliJ IDEA for Java and PyCharm for Python. It will compute internally the time each process takes to execute, by measuring the start and end time of each one. The results for each operation will be displayed on the graphical user interface created, so the user can compare and choose which progamming suits his needs the most.

1.3 Objectives

The objective of this project is to design and implement an application that computes the execution time of different processes, then compare the result obtained between three programming languages (C++, Java and Python). In order to do this I implement the operations in each programming language, then compute for them the start time and end time.

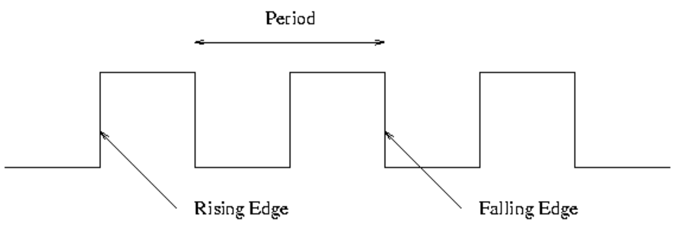
2. Bibliographic study

[1] **Measuring the performance** of a program means keeping track of the consumption of resources(such as execution time, reaction time, memory capacity etc.) used by the program.

Many methods are often not transferable to other platforms, which means that the way in which the execution time is measured might need to be adapted between platforms. Choosing the right method will depend on the operating system, programming language, and the technique. The computer I use for this application uses the operating system Windows 10 (version 21H2) and the system type is 64-bit operating system, x64-based processor (Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz).

When measuring time, Wall Time and CPU Time are often the ones computed. Wall Time represents the real-world time, elapsed from the start of an event to the end of it. CPU Time is represents the actual time spent by the CPU executing the process.

There are many methods for measuring the execution time of processes. The one used for this project would be the one of counting the clock. The CPU has an internal clock signal that goes from bottom 0 to up 1 (this is called rising-edge) or from 1 to 0 (this is called falling-edge) in microseconds (figure 1). Considering this, if one has the amount of clock cycles performed in a second and knows the difference of clock cycles calculated between two instants, that can be converted into seconds.

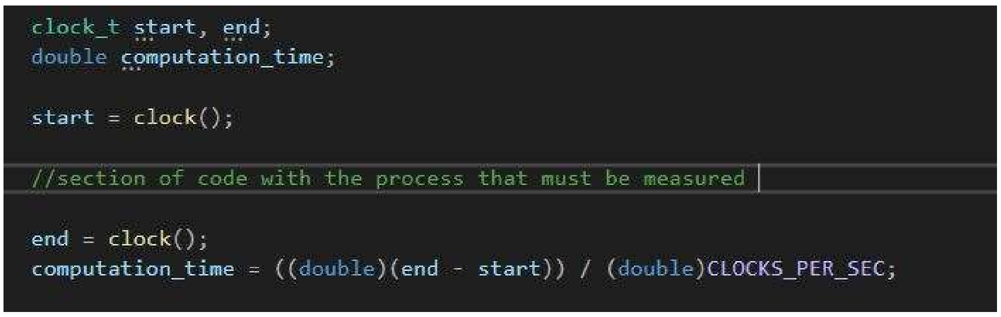


(figure1 – rising edge vs falling edge)

To measure the execution time, the measurement is made in two different time instants on the clock cycles passed from a certain start, to an end.

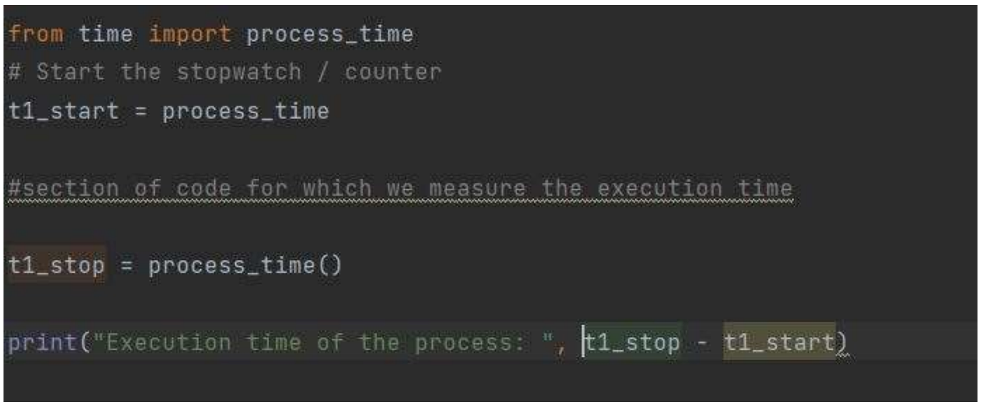
In C++, this can be done using the time utility, the clock() function and by knowing the CLOCKS_PER_SEC. By computing the difference between end and start and dividing by CLOCKS_PER_SEC, we obtain the seconds passed from the first call of clock() to the second call. (figure 2). It must be noted that CLOCKS_PER_SEC is not accurate to the thousandth of a second (by default the value of that variable is 1000000).

(figure 2 – computing execution time in C++)



```
clock_t start, end;
double computation_time;

start = clock();

//section of code with the process that must be measured

end = clock();
computation_time = ((double)(end - start)) / (double)CLOCKS_PER_SEC;
```

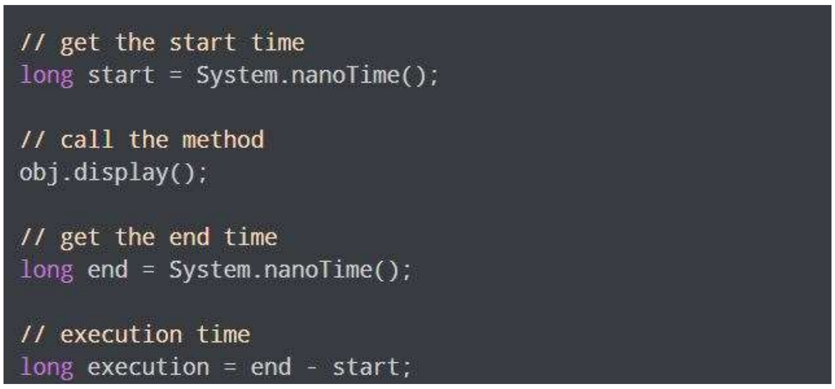
In Python, the execution time will be computed using the time.process_time() function. [2] This function returns the value (in fractional seconds) of the sum of the system and user CPU time of the current process, without including the time elapsed during sleep (to include the sleep time the pref_counter() function can be used). (figure 3)



```python
from time import process_time
# Start the stopwatch / counter
t1_start = process_time

#section of code for which we measure the execution time

t1_stop = process_time()

print("Execution time of the process: ", t1_stop - t1_start)
```

(figure 3 – computing execution time in Python)

In Java, to compute the elapsed time between the start and the end of a process, the method nanoTime() of the System class will be used. The nanoTime() method returns the current value of the running JVM in nanoseconds. [3]



```java
// get the start time
long start = System.nanoTime();

// call the method
obj.display();

// get the end time
long end = System.nanoTime();

// execution time
long execution = end - start;
```

(figure 4 – computing execution time in Java) [3]

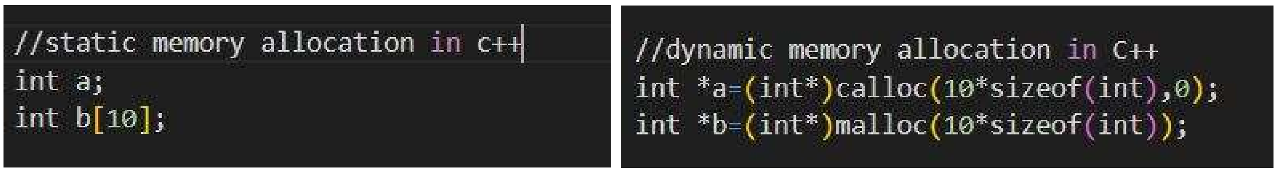
[4**] Memory allocation** is a process by which computer programs and services are assigned with physical or virtual memory space. It is the process of reserving a partial or complete portion of computer memory for the execution of programs and processes.

There are two types of memory allocation: static (the program is allocated memory at compile time) and dynamic (the programs are allocated with memory at run time).

[5] For the static type the compiler allocates the required memory for the program before the execution of the program. Since memory allocation takes place during compile time, It is also called compile-time memory allocation. No special functions are required for static allocation, just normally declare variables. Memory is allocated From Stack Memory.

[5] When memory for the program is allocated during execution time, it is called Dynamic Memory Allocation. The compiler allocates the required memory for the program during the execution of the program. Since memory allocation takes place during run time or execution time, It is also called run-time memory allocation. Memory is allocated on the heap

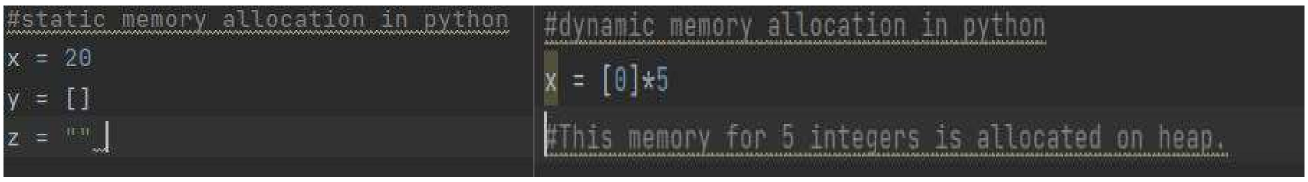
There is a difference between programming languages in how memory is allocated. Python and Java use a garbage collector, while C doesn't. Garbage collection is when the interpreter frees memory for your programs for you when it is not being used



```
//static memory allocation in c++
int a;
int b[10];
```

```
//dynamic memory allocation in C++
int *a=(int*)calloc(10*sizeof(int),0);
int *b=(int*)malloc(10*sizeof(int));
```

(figure 5 – difference between static and dynamic memory allocation in C++)

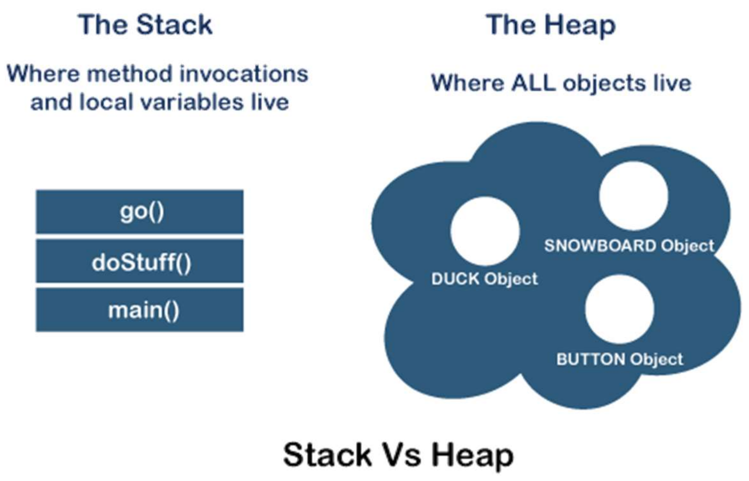


```
#static memory allocation in python
x = 20
y = []
z = ""
```

```
#dynamic memory allocation in python
x = [0]*5
#This memory for 5 integers is allocated on heap.
```

(figure 6 – difference between static and dynamic memory allocation in Python)

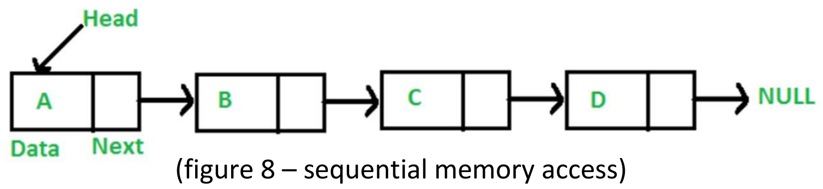
[6] In Java, the major difference between stack memory and heap memory is that the stack is used to store the order of method execution and local variables while the heap memory stores the objects and it uses dynamic memory allocation and deallocation.



(figure 7 – difference between static and dynamic memory allocation in Java)

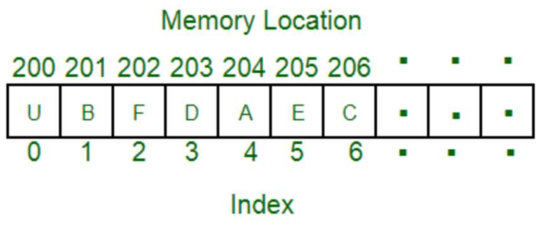
Memory access is the operation of reading or writing stored information. There are four types of memory access methods: sequential, random, direct and associate.

In the sequential access method, the memory is accessed in a specific linear sequential manner, like accessing in a single Linked List. The access time depends on the location of the data. [7]



(figure 8 – sequential memory access)

In the random access method, any location of the memory can be accessed randomly like accessing in Array. Physical locations are independent in this access method.



(figure 9 – direct/random memory access)

In the direct access method, individual blocks or records have a unique address based on physical location. access is accomplished by direct access to reach a general vicinity plus sequential searching, counting or waiting to reach the final destination. This method is a combination of above two access methods.

In associate access, a word is accessed rather than its address. This access method is a special type of random access method (used in cache memories).

A thread is a small set of instructions designed to be scheduled and executed by the CPU independently of the parent process.

[8] In C, a **thread is created** is created using the pthread_create() function, which takes four arguments: the first argument is a pointer to thread_id which is set by this function, the second argument specifies attributes (if the value is NULL, then default attributes shall be used), the third argument is name of function to be executed for the thread to be created and the fourth argument is used to pass arguments to the function.

[9] In Java, a thread can be created by implementing the Runnable interface and overriding the run() method. Then a Thread object can be created and the start() method called.

[10] The Python standard library provides threading. To start a separate thread in python, you create a Thread instance and then tell it to .start().

A **thread context switch** is the process of storing the state of a thread, so that it can be restored and resume execution at a later point, and then restoring a different, previously saved, state. This allows multiple processes to share a single central processing unit (CPU), and is an essential feature of a multitasking operating system. Context switches are usually computationally intensive, and much of the design of operating systems is to optimize the use of context switches. Switching from one process to another requires a certain amount of time for doing the administration. [11]

Thread migration in operating system terminology actually means moving the thread from one core's run queue to another. This is done by the operating system scheduler. [12]

3. Analisys

3.1. Project Proposal

The proposal for this project is to create an application that that runs three programs, written in three different progamming languages (Java, C, Python), in order to measure the execution time of the following processes: memory allocation, memory access (both static and dynamic), thread creation, thread context switch and thread migration. By doing this, the way in which each of the programming language does all of the processes earlier mentioned can be observed, most specifically the time it takes each programming language to execute these operations. Knowing which application is faster in executing a specific operation is important, because it can help in choosing the most efficient language for a given problem that requires that operation.

I will create three programs in each progamming language that will measure the execution time, each one in a different programming language, then I will have a main application, written in Java, which will call the executables of the previously mentioned programs.

The main application will have a graphical user interface, which will show the execution time of a process in each programming language.

3.2. Project Plan

I started working on the project since the second week of this semester. I made a plan for the following untill the deadline, for each week, in which I intend to:

- Week 2 : Choose of the project
- Week 3 : Write the introduction of the project (subject description context, where to use) and the bibliographic study.
- Week 4 : Present the introduction and the bibliographic study parts of the documentation
- Week 5 : Write the analisys part of the documentation (project proposal, plan, use case diagrams, functionality description)

- Week 6 : Present the analisys part of the documentatin
- Week 7 : Write the basic structure of the application, the design
- Week 8 : Present the progress of the documentation and of the application
- Week 9 : Work on the design and implementation of the application
- Week 10 : Present the progress of the documentation and the application
- Week 11: Test and validate the functionality of the application, by doing many measurements
- Week 12 : Present the documentation, the tests and the validation
- Week 13 : Write the conclusion and the bibliography in the documentation, finish the project
- Week 14 : Present the conclusions and the bibliography from the documentation
- Week 15 : Present the project: the application and final version of the documentation

3.3. Analisys

To measure the execution time of each process in different programming languages, I will create three applications, each one in a different programming language and in a different IDE (C in Visual Studio 2019, Python in PyCharm, Java in IntelliJ). The processes measured will be: memory allocation, memory access (both static and dynamic), thread creation, thread context switch and thread migration.

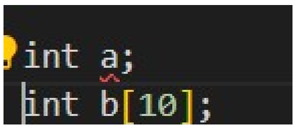
Each process, as well as the way in which we measure their execution time is done differently in each programming language.

The main program, in Java, will call the executable of each applcation and show the results on the graphical user interface.

3.3.1. **C**

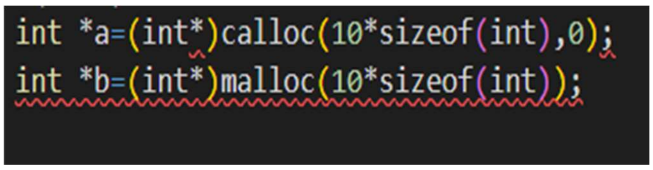
- Memory allocation:

No special functions are required for static allocation, just normally declare variables (figure 10) . Memory is allocated from stack memory. For dynamic memory allocation, functions calloc() and malloc() are used. (figure 11)



```
int a;
int b[10];
```

(figure 10 – static memory allocation in C)

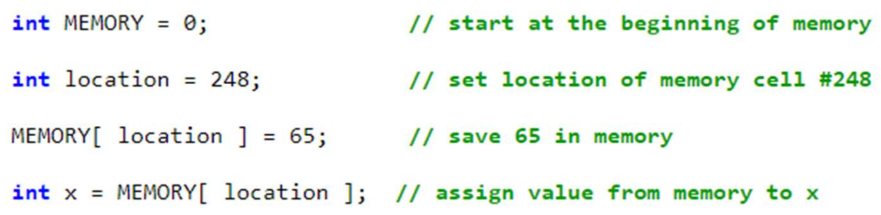


```
int *a=(int*)calloc(10*sizeof(int),0);
int *b=(int*)malloc(10*sizeof(int));
```

(figure 11 – dynamic memory allocation in

- Memory access

Because C is a system programming language, it must provide direct access to physical hardware, including memory [13]. In figure 13 there is an example of memory access in C.

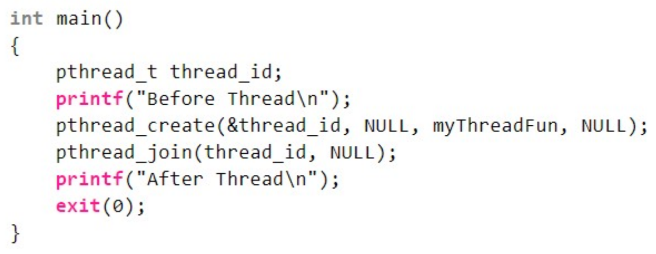


```c
int MEMORY = 0;                 // start at the beginning of memory

int location = 248;             // set location of memory cell #248

MEMORY[ location ] = 65;        // save 65 in memory

int x = MEMORY[ location ];  // assign value from memory to x
```

(figure 12 – memory access in C) [13]

- Thread creation

To create thread in C, the pthread functions are used (figure 13).



```c
int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
```

(figure 13 – thread creation in C) [8]

- Thread context switch

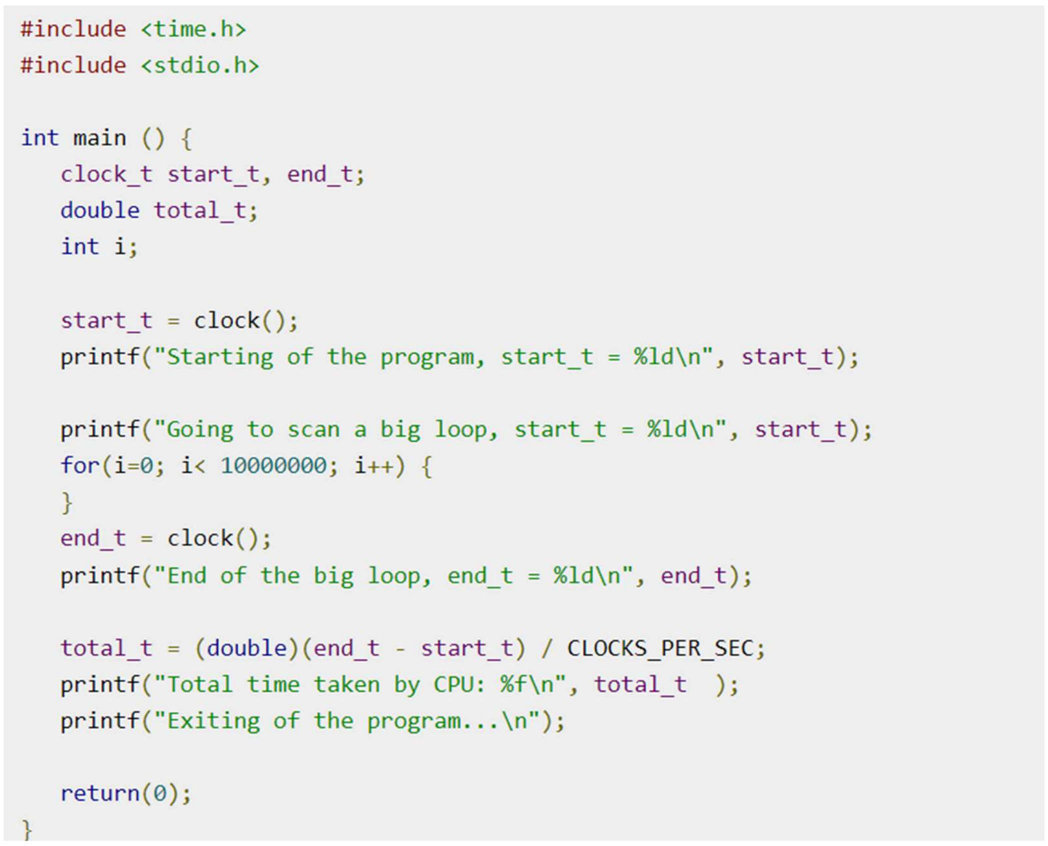
Thread context switching takes place when the CPU saves the current state of the thread and switches to another thread of the same process. [14] Context switching can be done by switching the context from one thread to another, using sincronization mechanisms like semaphores and locks.

- Thread migration

Thread migration (meaning moving the thread from one core's run queue to another) is done by the operating system scheduler.

Measuring Execution Time

For this project, for each operation I measure the CPU time. In C, this can be done by using the clock() function from the time.h library. This function returns the number of clock ticks elapsed since the start of the program. To obtain the final time, we need to divide the difference obtained using the clock function at the end and at the start of each operation and to divide it with CLOCKS_PER_SEC.(figure 14)



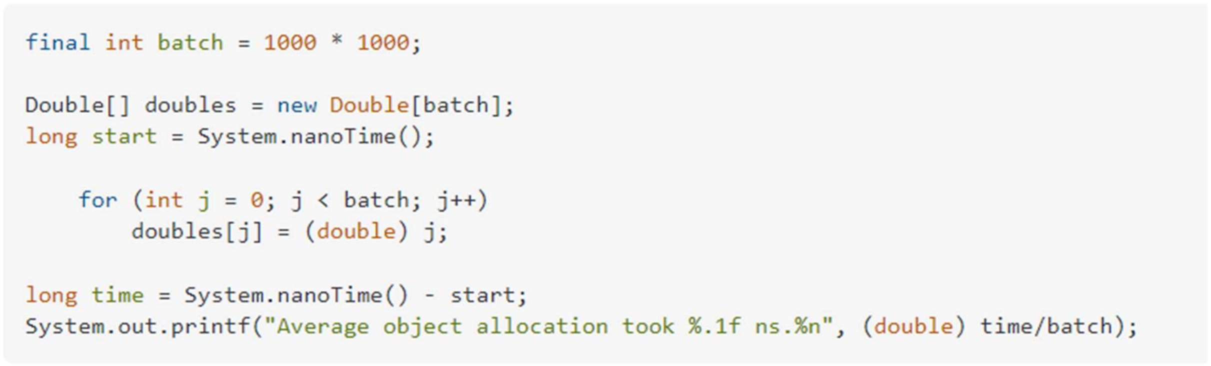
```c
#include <time.h>
#include <stdio.h>

int main () {
   clock_t start_t, end_t;
   double total_t;
   int i;

   start_t = clock();
   printf("Starting of the program, start_t = %ld\n", start_t);

   printf("Going to scan a big loop, start_t = %ld\n", start_t);
   for(i=0; i< 10000000; i++) {
   }
   end_t = clock();
   printf("End of the big loop, end_t = %ld\n", end_t);

   total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
   printf("Total time taken by CPU: %f\n", total_t  );
   printf("Exiting of the program...\n");

   return(0);
}
```

(figure 14 – computing CPU time in C) [15]

3.3.2. **Java**

- Memory allocation:

In Java, the stack is used to store the order of method execution and local variables while the heap memory stores the objects and it uses dynamic memory allocation and deallocation. Measuring the time it takes to allocate memory in Java can be done in a way like presented in figure 15.



```java
final int batch = 1000 * 1000;

Double[] doubles = new Double[batch];
long start = System.nanoTime();

    for (int j = 0; j < batch; j++)
        doubles[j] = (double) j;

long time = System.nanoTime() - start;
System.out.printf("Average object allocation took %.1f ns.%n", (double) time/batch);
```

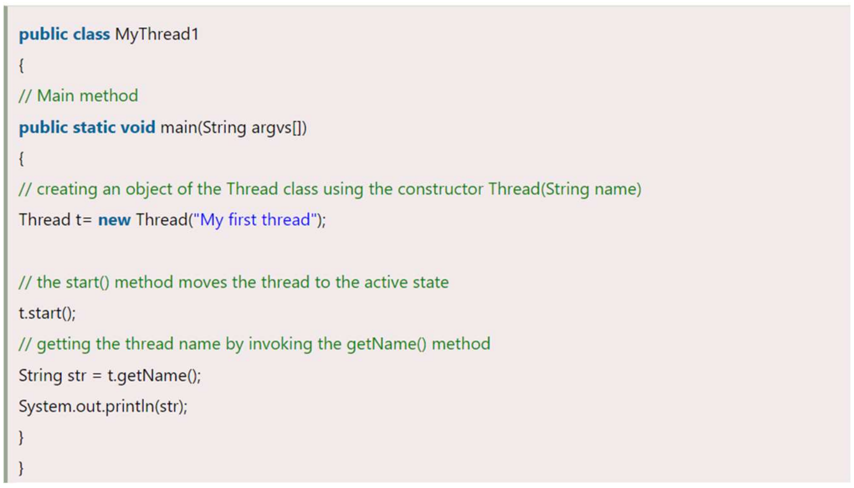
(figure 15 – computing time for memory allocation in Java) [16]

- Memory access

In java, there is an Unsafe class, which allows direct memory access, in a matter similar to C. Then the execution time can be measured like for any other operation.

- Thread creation

Thread class provide constructors and methods to create and perform operations on a thread.Thread class extends Object class and implements Runnable interface. (figure 16).



```
public class MyThread1
{
// Main method
public static void main(String argvs[])
{
// creating an object of the Thread class using the constructor Thread(String name)
Thread t= new Thread("My first thread");

// the start() method moves the thread to the active state
t.start();
// getting the thread name by invoking the getName() method
String str = t.getName();
System.out.println(str);
}
}
```

(figure 16 – creating threads in Java) [9]

- Thread context switch

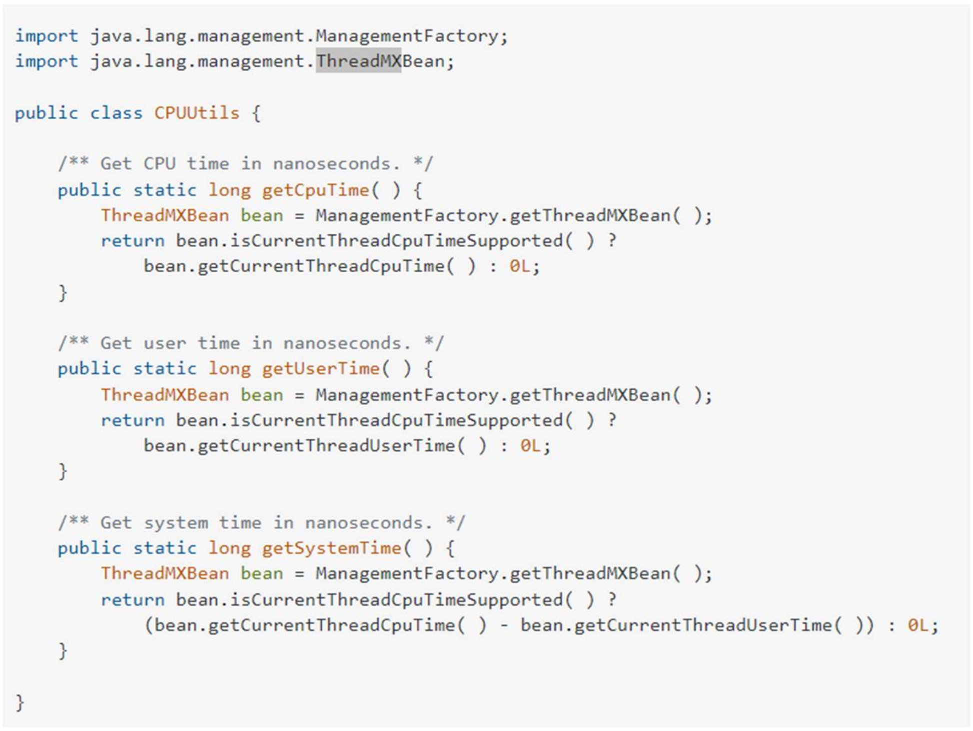
Thread context switching in Java can be done by using function like sleep(), to pause one thread so another one can continue, using sinchronization mechanism, or using a combination of wait() and notify() to carry out inter-thread communication

- Thread migration

Java Thread objects delegate to the OS's thread management system, so the exact behavior depends on the OS.

- Measuring Execution Time

In Java, ThreadMXBean.getThreadCPUTime() is the function used to find out how much CPU time a given thread has used. Use ManagementFactory.getThreadMXBean() to get a ThreadMXBean and Thread.getId() to find the id of the thread we want to compute the time for. (figure 17)



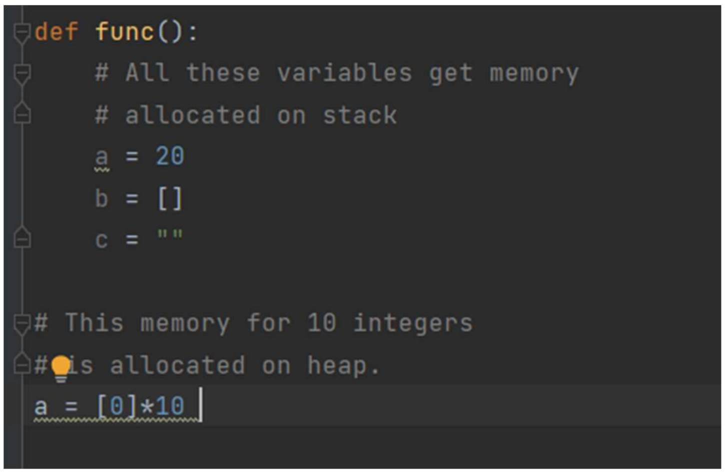
```java
import java.lang.management.ManagementFactory;
import java.lang.management.ThreadMXBean;

public class CPUUtils {

    /** Get CPU time in nanoseconds. */
    public static long getCpuTime( ) {
        ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
        return bean.isCurrentThreadCpuTimeSupported( ) ?
            bean.getCurrentThreadCpuTime( ) : 0L;
    }

    /** Get user time in nanoseconds. */
    public static long getUserTime( ) {
        ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
        return bean.isCurrentThreadCpuTimeSupported( ) ?
            bean.getCurrentThreadUserTime( ) : 0L;
    }

    /** Get system time in nanoseconds. */
    public static long getSystemTime( ) {
        ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
        return bean.isCurrentThreadCpuTimeSupported( ) ?
            (bean.getCurrentThreadCpuTime( ) - bean.getCurrentThreadUserTime( )) : 0L;
    }

}
```

(figure 17 – compute CPU time in Java) [17]

3.3.3. **Python**

- Memory allocation:

In python, the methods/method calls and the references are stored in stack memory and all the values objects are stored in a private heap.

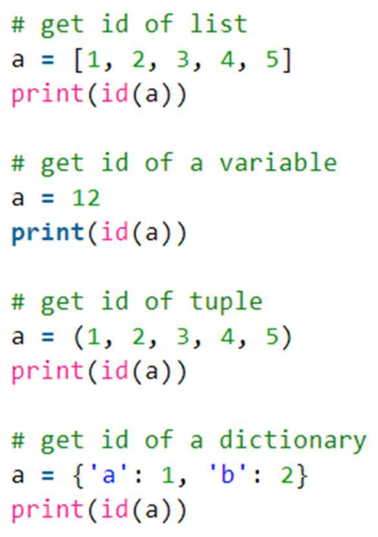


```python
def func():
    # All these variables get memory
    # allocated on stack
    a = 20
    b = []
    c = ""

# This memory for 10 integers
# is allocated on heap.
a = [0]*10
```

(figure 18 – memory allocation in python)

- Memory access

In python, in order to access memory, we can get an address using the id() function. id() function gives the address of the particular object.

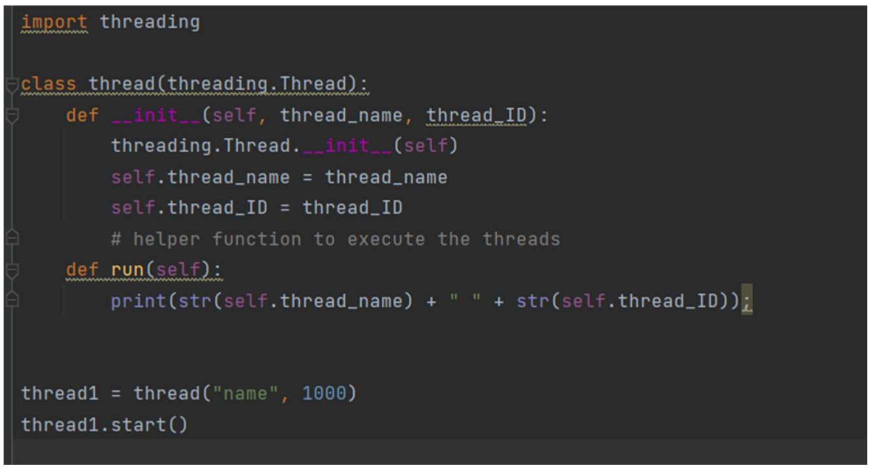


```python
# get id of list
a = [1, 2, 3, 4, 5]
print(id(a))

# get id of a variable
a = 12
print(id(a))

# get id of tuple
a = (1, 2, 3, 4, 5)
print(id(a))

# get id of a dictionary
a = {'a': 1, 'b': 2}
print(id(a))
```

(figure 19 – memory access address in python) [18]

- Thread creation

In python, a thread can be created by importing the threading module, and then creating a thread class. (figure 20).



```python
import threading

class thread(threading.Thread):
    def __init__(self, thread_name, thread_ID):
        threading.Thread.__init__(self)
        self.thread_name = thread_name
        self.thread_ID = thread_ID
        # helper function to execute the threads
    def run(self):
        print(str(self.thread_name) + " " + str(self.thread_ID));


thread1 = thread("name", 1000)
thread1.start()
```

(figure 20 – creating threads in python)

- Thread context switch

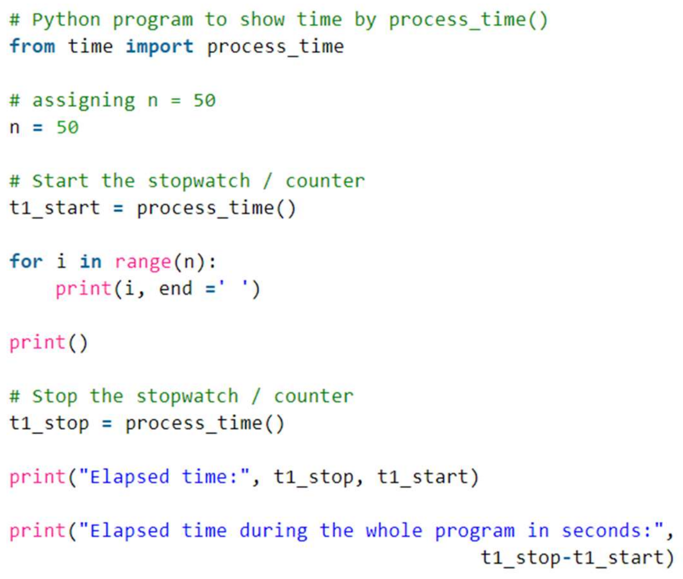
In Python 3.2 the interpreter's thread switch time interval can be adjusted the function sys.setswitchinterval and some sinchronisation mechanism

- Thread migration

Thread migration in Python is done in a similar way as in the other two programming languages (is done by the operating system scheduler.)

- Measuring Execution Time

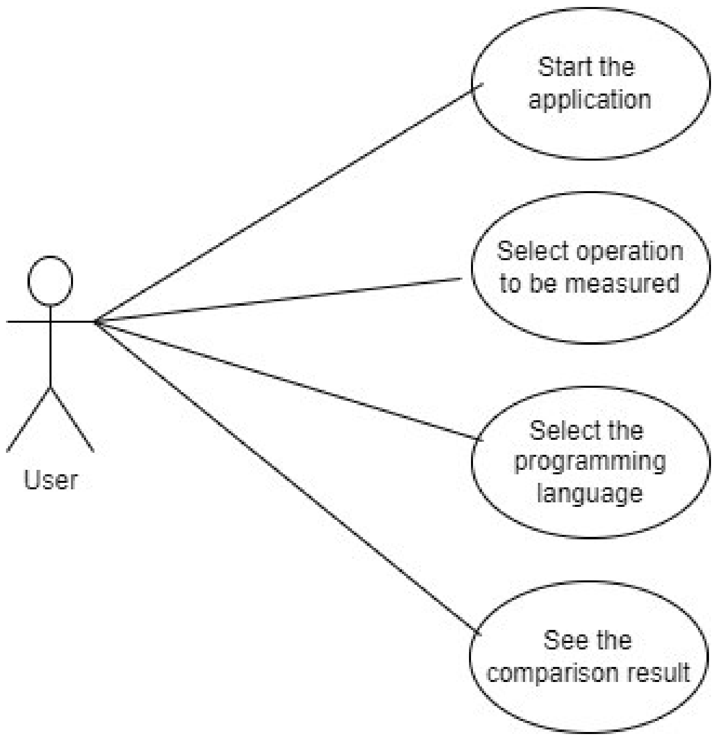
In Python, to measure the CPU time of each process, I use the function time.process_time(), which return the value (in fractional seconds) of the sum of the system and user CPU time of the current process. (figure 21)



```
# Python program to show time by process_time()
from time import process_time

# assigning n = 50
n = 50

# Start the stopwatch / counter
t1_start = process_time()

for i in range(n):
    print(i, end =' ')

print()

# Stop the stopwatch / counter
t1_stop = process_time()

print("Elapsed time:", t1_stop, t1_start)

print("Elapsed time during the whole program in seconds:",
                                         t1_stop-t1_start)
```

(figure 21 – measure CPU time in python)[19]

3.4. Use case diagram


Start the application
Select operation to be measured
Select the programming language
See the comparison result
User

4. Design
5. Implementation
6. Testing and validation
7. Conclusions

8. Bibliography

[1] „Measuring the performance” [Online]. Available: https://serhack.me/articles/measure-execution-time-program/

[2] „time.process_time() function in Python” [Online]. Available: https://www.geeksforgeeks.org/time-process_time-function-in-python/

[3] ”Execution time in Java” [Online]. Available: https://www.programiz.com/java-programming/examples/calculate-methods-execution-time

[4] ”Memory Allocation” [Online]. Available: https://www.techopedia.com/definition/27492/memory-allocation

[5] ”Static and dynamic memory allocation” [Online]. Available: https://www.codingninjas.com/codestudio/library/difference-between-static-and-dynamic-memory-allocation-in-c

[6] ”Memory management in Java” [Online]. Available: https://www.javatpoint.com/memory-management-in-java

[7] ”Memory Access Methods” [Online]. Available: https://www.geeksforgeeks.org/memory-access-methods/

[8] ”Thread in C” [Online]. Available: https://www.geeksforgeeks.org/multithreading-c-2/

[9] „How to create a thread in Java” [Online]. Available: https://www.tutorialspoint.com/how-to-create-a-thread-in-java

[10] ”Threading in Python” [Online]. Available: https://realpython.com/intro-to-python-threading/

[11] ”Context switch” [Online]. Available: https://en.wikipedia.org/wiki/Context_switch

[12] ”Thread migration” [Online]. Available: https://www.quora.com/What-is-thread-migration-Should-the-thread-image-be-the-same-moving-from-one-process-to-another

[13]”Memory access in C”[Online]. Available: http://www.c-jump.com/bcc/c155c/MemAccess/MemAccess.html

[14]”Thread Context Switch”[Online]. Available: https://levelup.gitconnected.com/know-the-difference-between-process-and-thread-context-switching-edce0c6a47f4

[15]”Compute CPU time in C”[Online]. Available: https://www.tutorialspoint.com/c_standard_library/c_function_clock.htm

[16]”Compute time for memory allocation in Java”[Online]. Available: https://stackoverflow.com/questions/8648826/what-is-the-typical-speed-of-a-memory-allocation-in-java

[17]”CPU time in Java”[Online]. Available: https://stackoverflow.com/questions/180158/how-do-i-time-a-methods-execution-in-java

[18]”Memory access in Python”[Online]. Available: https://www.geeksforgeeks.org/how-to-get-the-memory-address-of-an-object-in-python/

[19]”CPU time in Python”[Online]. Available: https://www.geeksforgeeks.org/time-process_time-function-in-python/