Department of Computer Science
Technical University of Cluj-Napoca

Structure of Computer Systems

Laboratory activity

Programming Language Racer

Project no. 37

Name:
Victor Moșolea

Group:
30432

Email:
mvictorandrei@gmail.com

Contents

Chapter 1

Introduction

1.1 Context

The goal of this project is to put programming languages up against each other. The categories which have been compared are: memory access (static, dynamic or both depending on the programming language), thread creation, and thread context switching. These categories were mentioned in the project specification. I decided to add another category in which we implement radix sort to really see who performs best.

1.2 Objectives

The objective for the project is to see which programming language out of C/C++, Java and Python performs best in each category and best overall.

1.3 Specifications

It is worth mentioning the specifications of all the used tools and the system specifications that were used for making the measurements as the results will differ based on the system:

- OS: Ubuntu 20.04
- Processor: Intel® Core™ i7-8665U CPU @ 1.90GHz × 8 64-bit
- 15,5 GB RAM
- gcc/g++ version 9.3.0
- OpenJDK 11.0.7
- Python 3.8.10

Unfortunately, the C/C++ programs have been written using the POSIX interface in order to allocate system resources such as threads, locks or semaphores. Therefore they will only work on UNIX systems. However, for the Python and Java programs this will not be an issue.

Chapter 2

Analysis

2.1 Memory access

Memory access is the operation of reading or writing stored information. There are two basic types of memory allocation: **static** and **dynamic**.

Out of the 3 benchmarked languages only C/C++ implements both static and dynamic memory allocation while Python and Java have only dynamic allocation.

What was measured? All programs have an array of integers stored in memory and have to access 500 of those elements randomly and that is what we measure: **time taken for 500 accesses**.

2.2 Threads

In computer science, a **thread** of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

2.2.1 Thread creation

Processes can ask the operating system to allocate new threads for them.

What was measured? Each program creates 100 threads which don't do anything and exit.

2.2.2 Thread context switch

Thread switching is a type of context switching from one thread to another thread in the same process.

What was measured? For this measurement I decided to create two threads which implement producer-consumer and just change between each other as fast as possible. Each program was given 1 second to do as many thread switches as possible.

2.3 Radix sort

Now for the grand finale, we check which language truly performs the best. Each program is given an array of integers and it has to perform radix short. The shorter the time means the better the performance in the end.

Chapter 3

Usage

I have also prepared a script that will make it easier for us to compile all the files, run them and compare the results.

3.1 Compiling the files

In order to compile the files we must run the command:

python3 programming_language_racer.py –prepare

After that everything we will be all ready to go.

3.2 Running the tests

There are multiple ways we can run the tests depending on what we want to see. By default the number of tests is 10, meaning that each program will be run 10 times. In order to test all categories we can run:

python3 programming_language_racer.py –race all

We can also test only some of the categories by running a command such as (in this case only radix sort is tested):

python3 programming_language_racer.py –race radix_sort

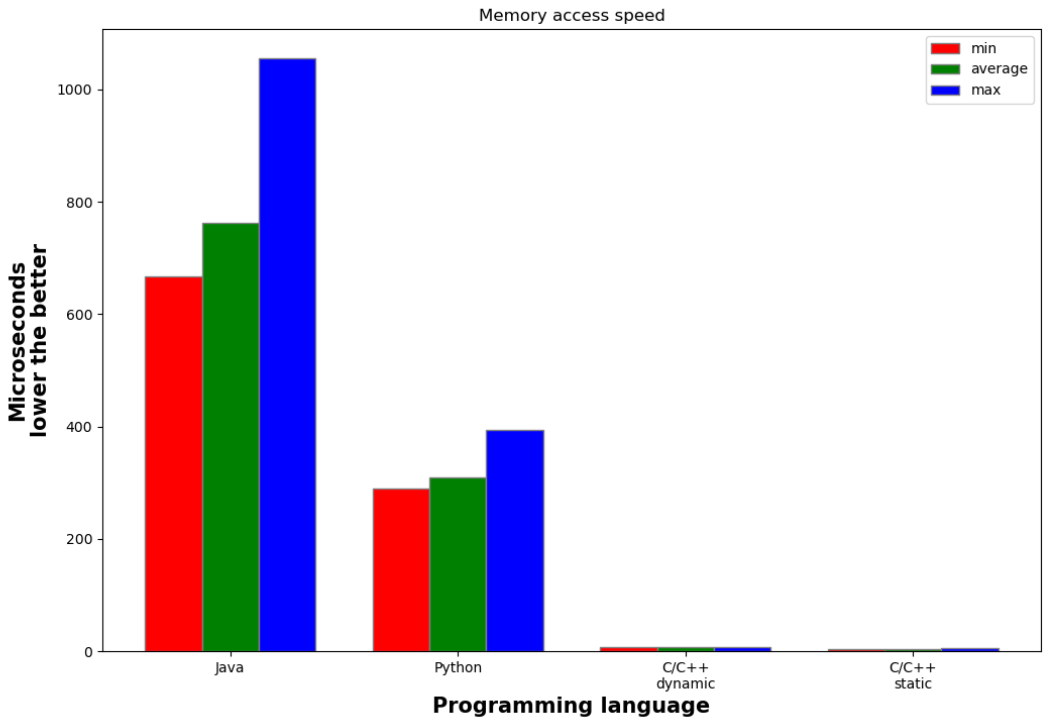
You can read more about the usage of this script by just running:

python3 programming_language_racer.py -h

Chapter 4

Interpretation of results

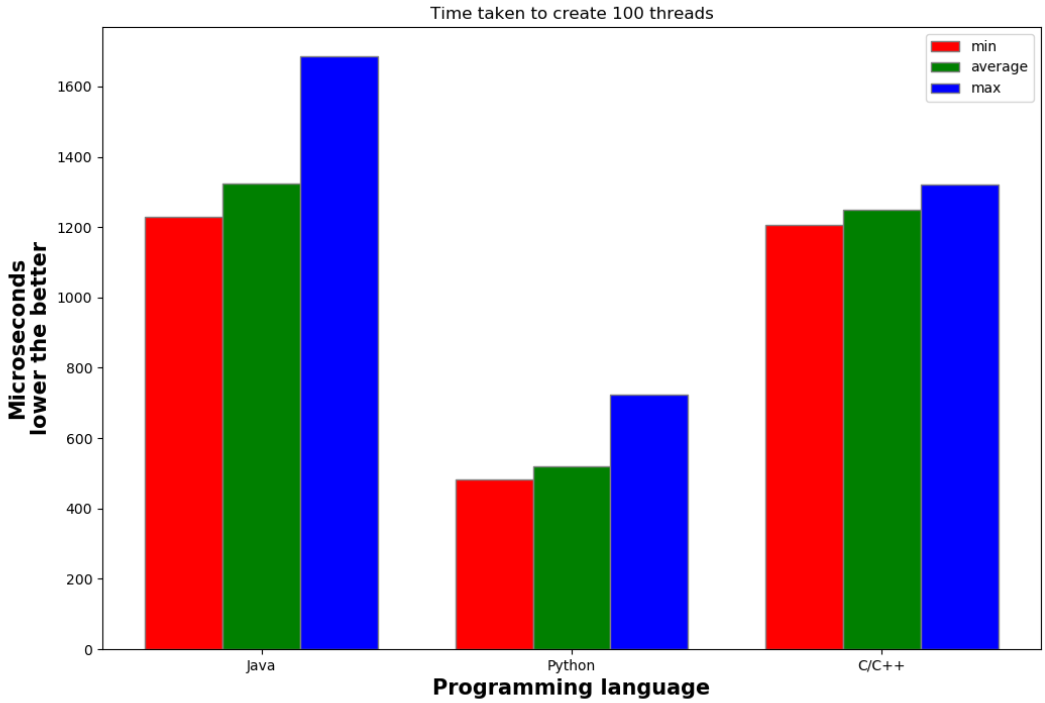
4.1 Memory access



.

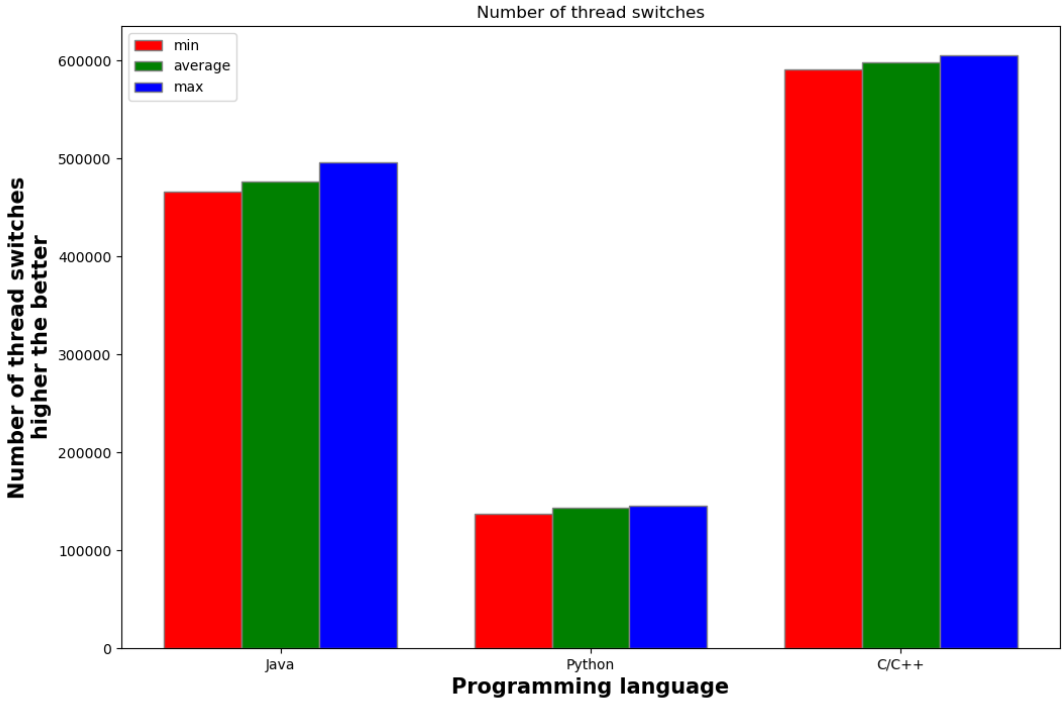
As shown in the graph, C/C++ is superior by far to both Java and Python in this aspect. Static memory access is slightly better than dynamic memory access for C/C++.

4.2 Thread creation



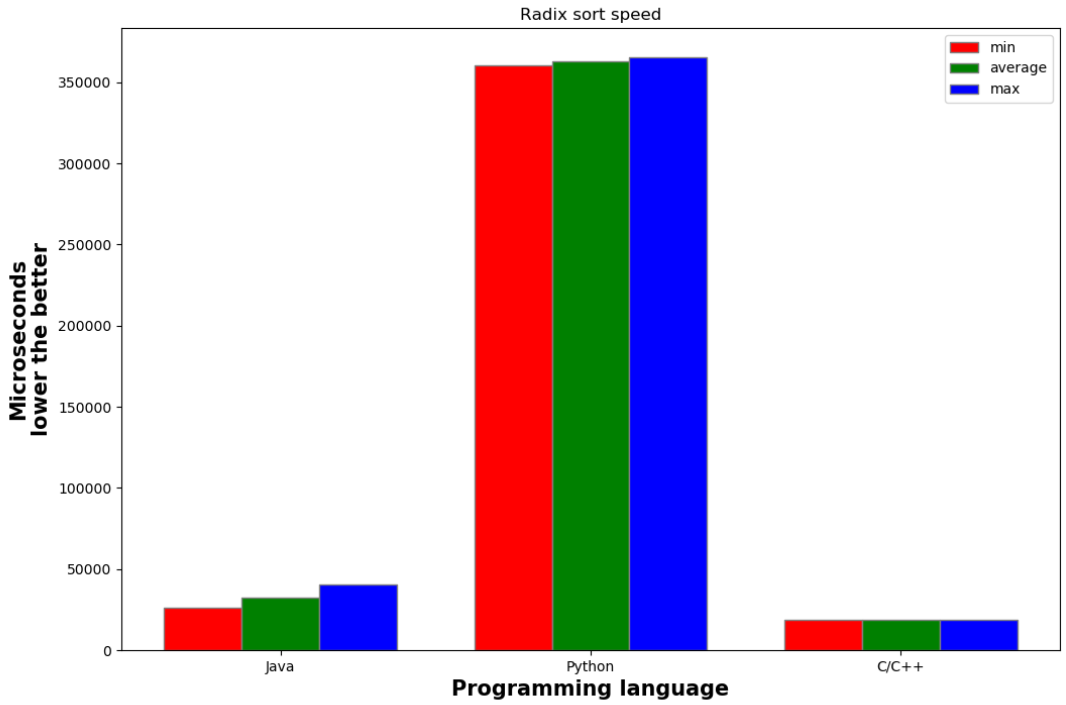
Although not known for its performance, Python wins this one while C/C++ comes in second and Java in third.

4.3 Thread context switch



Again, C/C++ wins a category as it does close to 600k thread context switches per second, while Java is just below 500k and Python below 200k.

4.4 Radix sort



.

As said previously, this is the ultimate benchmark for performance as it involves a lot of calculations and memory access. The result remain mostly the same:

1. C/C++
2. Java
3. Python

Chapter 5

Conclusions

The goal of this project was to measure programming languages up against each other and I think it achieved that. The results are not really surprising as C/C++ is known to be one of the best performers and these results confirmed that. All in all, I enjoyed this project a lot as it involved some of my favorite programming languages and I think I learned a lot of new things both about programming and hardware.

Bibliography

1. https://stackoverflow.com/
2. https://www.gnu.org/software/libc/manual/html_mono/libc.html
3. https://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html
4. https://docs.python.org/3/library/threading.html