A Case Study on Performance Optimization Techniques in Java Programming

Cirpian Khlud[1][a] and Cristian Frăsinaru[1][b]

[1]*Institute of Problem Solving, XYZ University, My Street, MyTown, MyCountry*
[2]*"Alexandru Ioan Cuza University", Iaşi, Romania*
ciprian.mustiata@gmail.com, acf@info.uaic.ro

Abstract: Choosing the right programming platform for processor or memory intensive applications is a subject that is debated in all types of contexts. When analyzing the performance of a specific platform, equally important is the usage of appropriate language specific constructions and programming interfaces (APIs). In this paper we investigate how a state-of-the art implementation, part of a multi-threaded framework for sequence analysis (elPrep) could benefit from various optimization techniques dedicated to improving the runtime performance of Java applications. ElPrep is an established tool for processing SAM and BAM files in sequencing pipelines. In order to migrate from its original implementation to a different software platform, more suitable for memory intensive tasks, the authors have reimplemented elPrep in Java, Go and C++ and benchmarked their performance. Overall, the Go implementation won by a good margin, considering a metric that involved both the RAM usage and the runtime performance. We show that, without changing the semantics of the algorithm, by using appropriate programming techniques we are able to significantly improve the behavior of the Java implementation, to a point that may even alter the conclusions of the original study.

1 INTRODUCTION

Here we cite the original paper (Costanza et al., 2019).

```
System.out.println("Hello World!!");
```

REFERENCES

Costanza, P., Herzeel, C., and Verachtert, W. (2019). Comparing ease of programming in c++, go, and java for implementing a next-generation sequencing tool. *Evolutionary Bioinformatics*, 15:1176934319869015.

[a] https://orcid.org/0000-0000-0000-0000
[b] https://orcid.org/0000-0002-5246-7396