# Report List-based sets

Author: Barau Elena-Raluca, Marculescu Tudor

# Table of Contents

# Introduction

The goal of this project is to get familiar with optimistic fine-grained locking schemes in concurrent data structures. Our running example is sorted linked list used to implement a set abstraction. The project was conducted in the context of "Fondements des algorithmes répartis (partie A)" course.

It proves that the performance is affected by the synchronization methods used on shared objects. The purpose of this project is to show how different ways of locking a set based list relates to the performance measured in a multi-threaded environment.

The hardware environment used was a virtual machine from the infrastructure of the faculty, namely: **lame21.enst.fr** which has the following specifications:

- Operating System: Ubuntu 24.04.3 LTS

   Kernel: Linux 6.8.0-55-generic

- Processor: 2x Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz

   Architecture: x86-64

   Per processor cores: 10 and threads: 2 per core, in total 20

   Total cores 20, total threads 40

- Memory: 256 GiB

- Storage: HDD 256 GiB

The software environment used to run synchrobench, which is the java based software suite to test the performance of our shared objects is:

- GNU Make v4.3

- Apache Ant v1.9.5

- Java jdk v1.7.0

- synchrobench release v1.1.0-alpha

Please notice that Java 8 caused issues when compiling the synchrobench project, particularly when the runtime jar was instrumented to enable transactional memory. We resorted to fallback to Java 7 in order to prevent this issue.

# Hand-over-Hand algorithm

The inspiration of implementing this algorithm came from the book "The art of multiprocessor programming" by Maurice Herlihy and Nir Shavit. In order to not start from scratch, the CoarseGrainedListBasedSet file was taken as an example and have been heavily modified in order to introduce the necessary logic of implementation for Hand-over-Hand algorithm.

The methods that have been modified are: addInt, removeInt and containsInt. Previously, a lock was used in order to prevent the access of the other threads to the whole object, which is made obsolete by the Hand-over-Hand algorithm which uses a lock on each Node of the list. Therefore the Node class was also modified to accommodate this change.

# Correctness of Hand-over-Hand algorithm

[] Argue that both safety and liveness hold

# Performance analysis

For the performance analysis, prepare a graph depicting the throughput as a function of the number of threads for the three algorithms, for some representative list size and update ratio. You may use gnuplot or any other plotting program of your choice. Then, for each algorithm, fix the update ratio to 10%, and prepare a graph depicting the throughput as the function of the number of threads, varying the list size. Finally, for each algorithm, prepare a graph depicting the throughput as the function of the number of threads, varying the update ratio, for the list size 1k. Altogether, this gives:

[] Three plots (one per algorithm), with three curves each, for a fixed update ratio 10% and varying list size.

[] Three plots (one per algorithm), with three curves each, for a fixed list size 100 and varying update ratios.

[] One plot, with three curves (one per algorithm), with fixed update ratio 10% and list size 1000.

[] Explain the forms of the curves and their relations to each other.

# Conclusions

# References