Parallel Processing Comparison in Computing Languages

A comparison between software parallelization in C, Go and Python

Feras Alshehri
CS 259 - Parallel processing
Summer 2021
San Jose State University

Outline

Brief

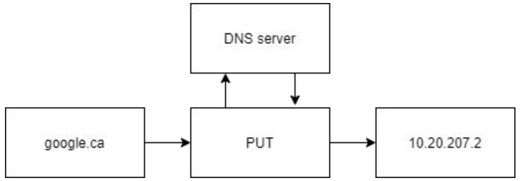
- The goal of this presentation demonstrate the impact of parallelization of the same program in different languages.
- The main profiling criteria will be the overall time of execution it takes the program to complete the same job.

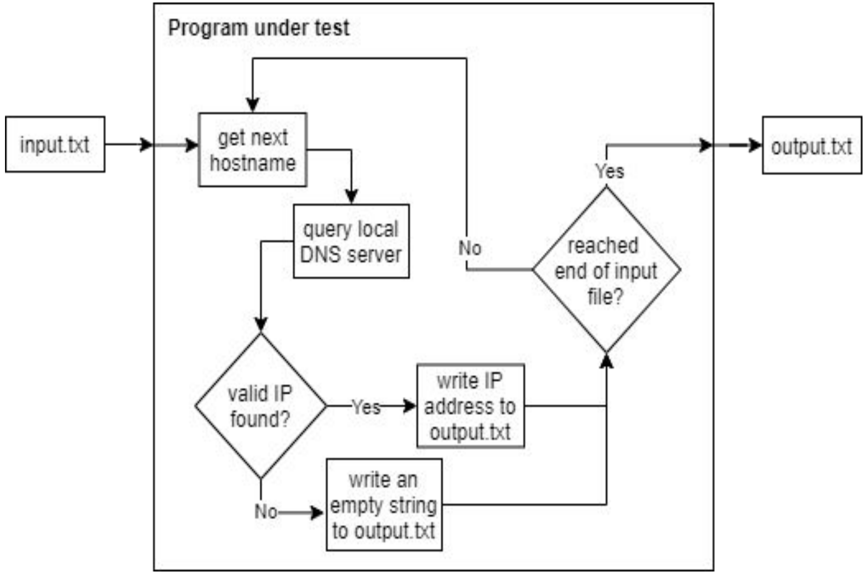
Experiment Design: Overview

- Experiment requirements:
    - Software requirements:
        - The experiment must be bounded by some external factor that can be overcome by a software parallelization mechanism (e.g., multithreading).
        - Parallelized PUT must protect shared resources to maintain original functionality.
    - Hardware requirements:
        - Multi-core CPU to allow adoption of each of the language's available parallelization methods.

Experiment Design: Programs Under Test (PUT)

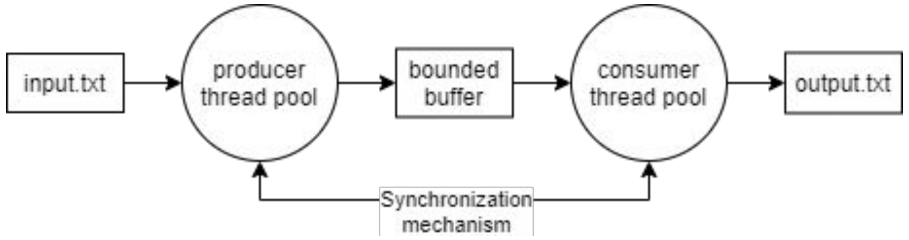
- PUT is fed a hostname as an input.
- PUT queries the nearest DNS server to get the IP address corresponding to the given hostname.
- PUT outputs the received IP address.





Experiment Design: Parallelization Opportunities

- PUT, in its sequential implementation, is limited by the duration of the DNS server query, and the speed at which we can read and write to the input and output files.
- To overcome this limit, we can employ the producer-consumer model.
- We can read more hostnames from the input file, or write a returned IP address to the output file, while we wait on the DNS query to return.



Experiment Design: Test Automation

- Handles executing each of the PUT given, with the correct input arguments, while allowing for additional iterations of each execution.
  - Additional iterations can be used to in data processing to remove the outliers.
- Manage profiling data format and location.
  - Total hostnames handled by PUT.
  - Total execution time.

Experiment Design: Data Processing Automation

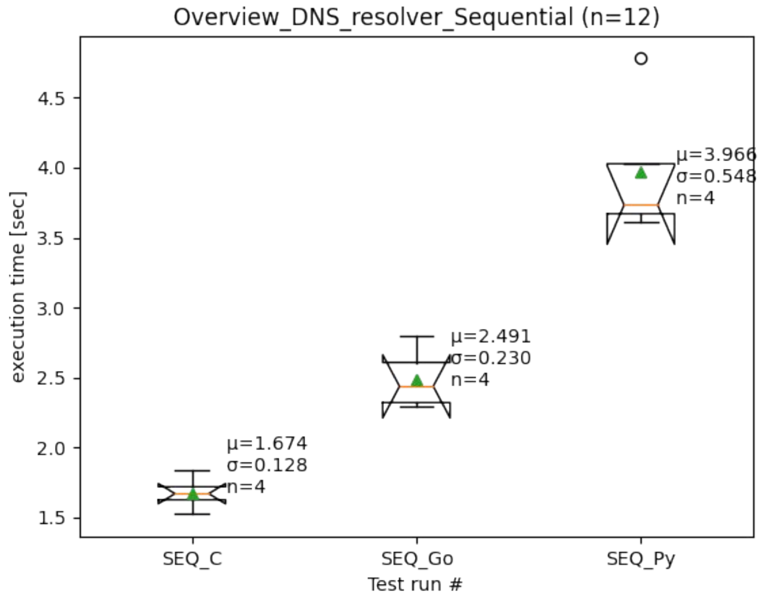
- Parse profiling data as generated by the test automation tool.
- Calculate statistics of each PUT’s execution runs.
- Generate raw statistics data to be available for further analysis.
- Generate plots for easier visualization.

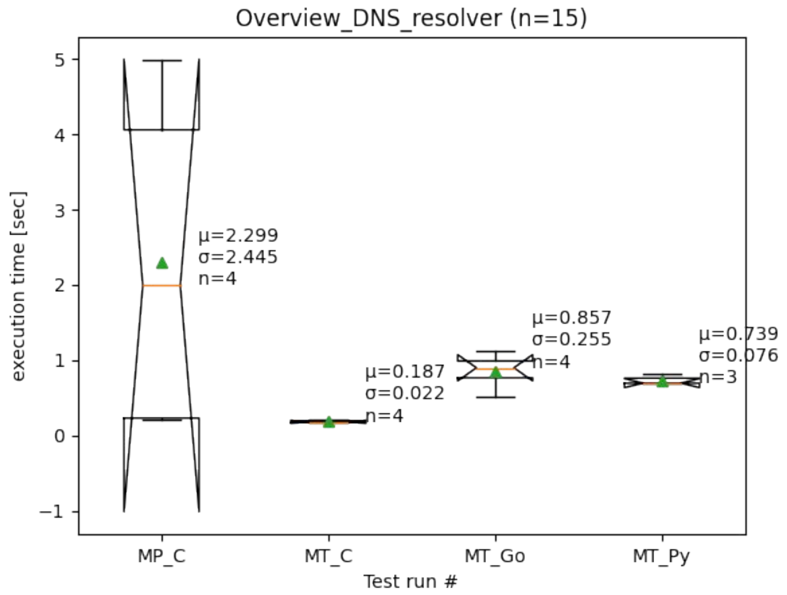
Results

- The difference between each PUT’s sequential and parallel version is visible with 200 hostnames.
  - It’s even more visible with 2000 hostname, and will be more visible with more hostnames.
- The percentage of improvement between each language’s PUT varies.
- In all of the following plots:
  - n = number of data points (complete runs)
  - μ = arithmetic mean
  - σ = standard deviation

Results: Sequential vs Parallelized PUT (overview)

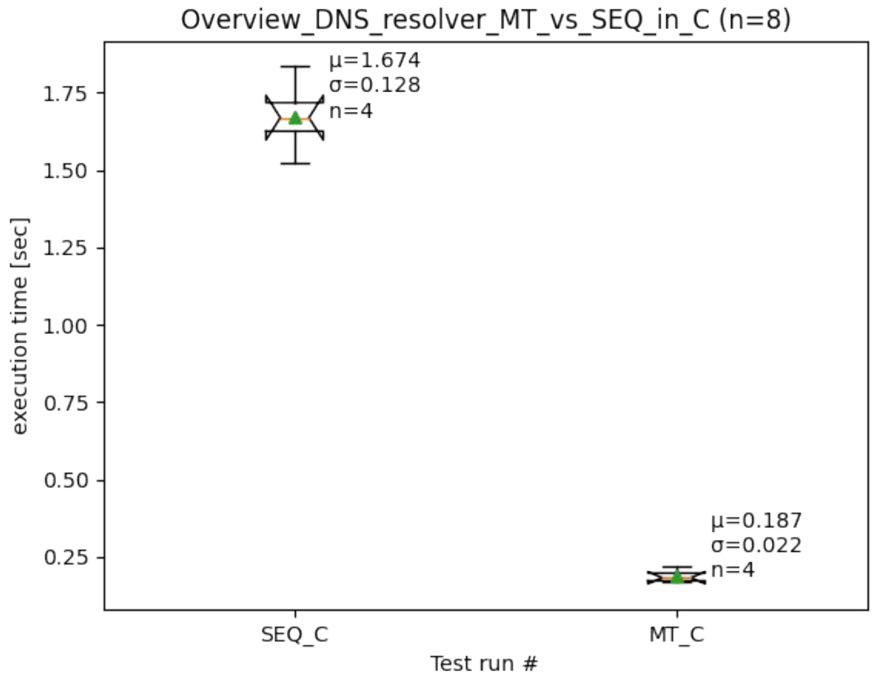
102 hostnames, 5 input files, ~20 hostnames per file





Results: C

102 hostnames, 5 input files, ~20 hostnames per file

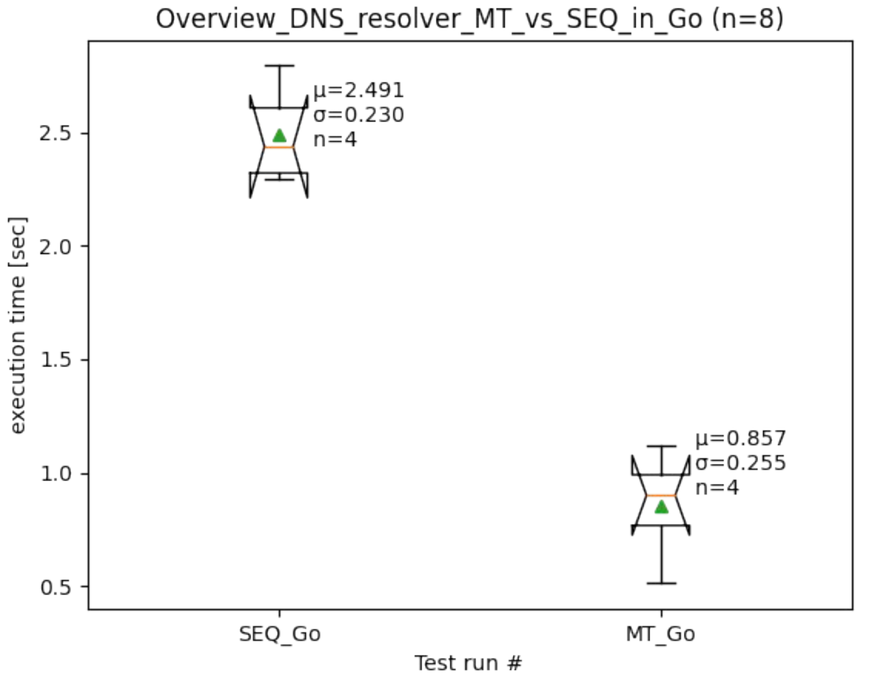


Parallelization in C yielded a decrease in total execution time of

88.829%

Results: Go

102 hostnames, 5 input files, ~20 hostnames per file

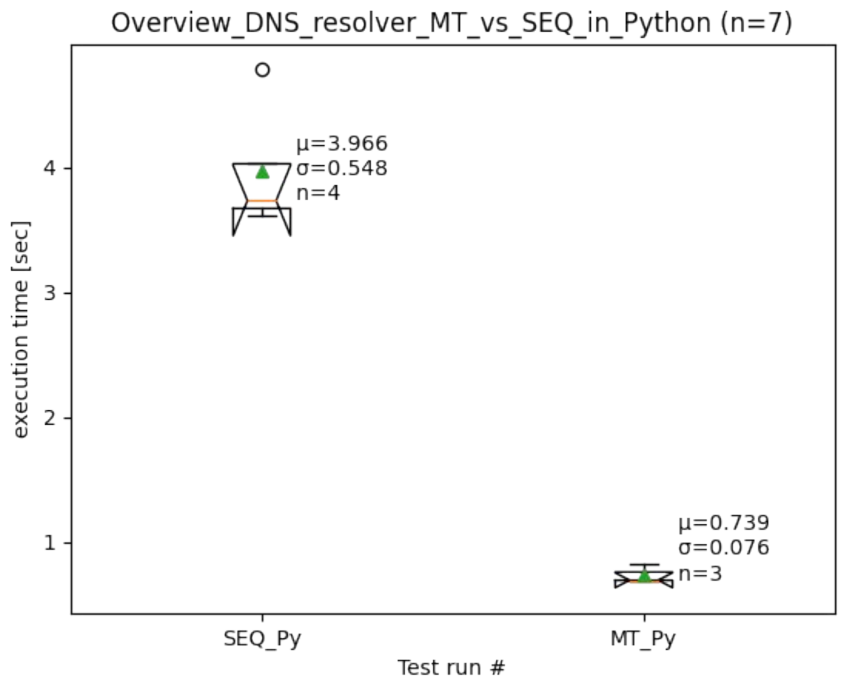


Parallelization in Go yielded a decrease in total execution time of

65.596%

Results: Python

102 hostnames, 5 input files, ~20 hostnames per file

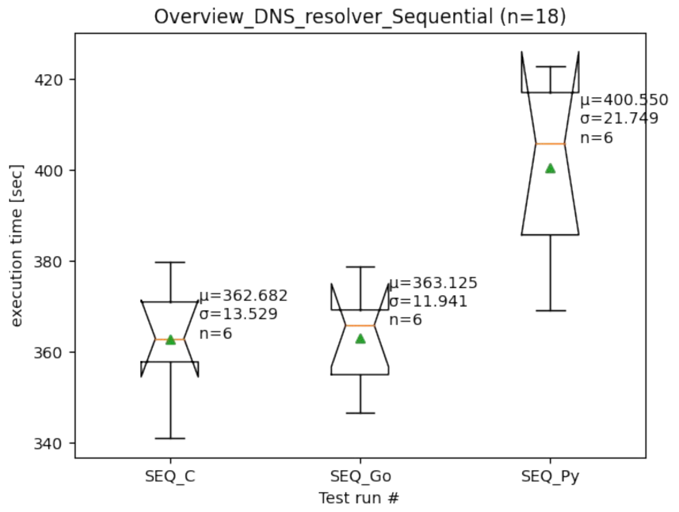


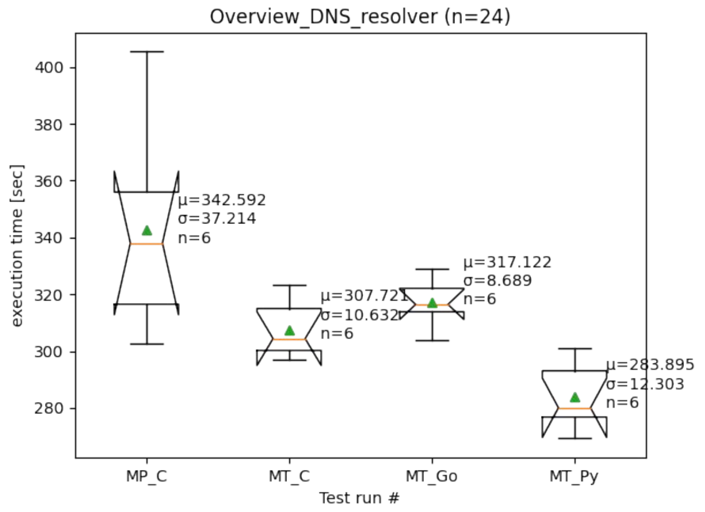
Parallelization in Python yielded a decrease in total execution time of

81.367%

Results: Sequential vs Parallelized PUT (overview)

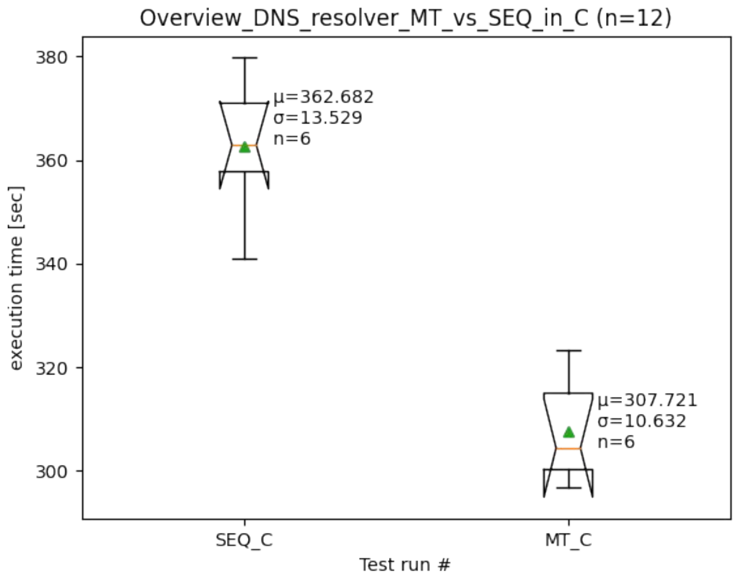
2000 hostnames, 4 input files, 500 hostnames per file





Results: C

2000 hostnames, 4 input files,
500 hostnames per file

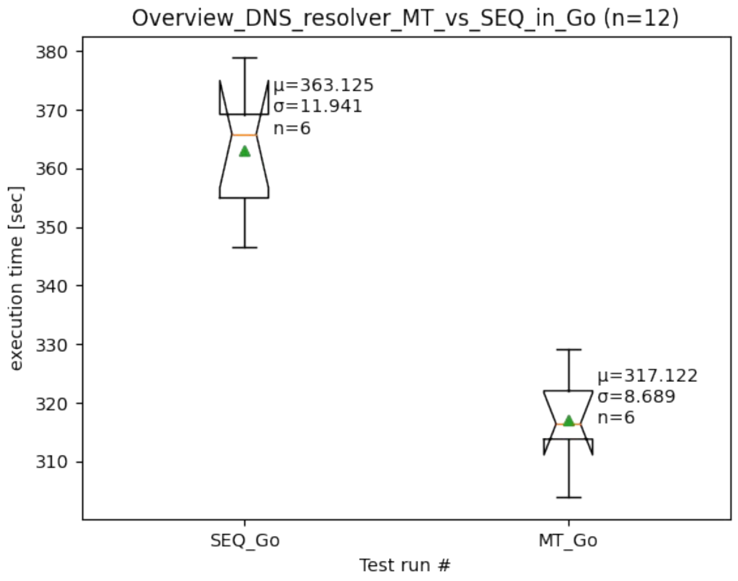


Parallelization in C yielded a decrease in total execution time of

15.154%

Results: Go

2000 hostnames, 4 input files,
500 hostnames per file

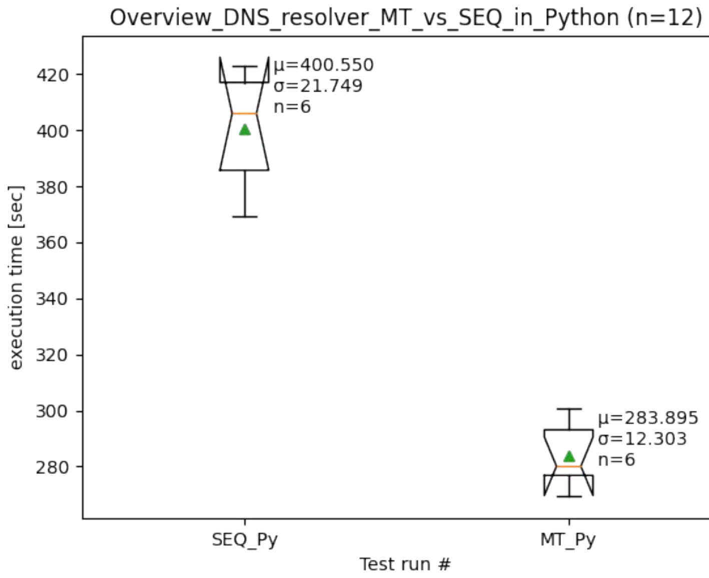


Parallelization in Go yielded a decrease in total execution time of

12.669%

Results: Python

2000 hostnames, 4 input files,
500 hostnames per file



Parallelization in Python yielded a decrease in total execution time of

29.124%

Reflection

A take away from this study, and how to apply it in the real-world

When planning to improve existing and established sequential software, and once software parallelization has been deemed the best path forward, it is worth the consideration to evaluate the options of porting the software to another development language and environment. This will most likely be limited by the dependency and integration of said software.

Challenges/Improvements

- Available hardware on hand is equipped with a dual-core processor and limited RAM memory (4GB).
  - This limits the visible impact of software parallelism.
- Software development and testing within the timeline at hand.

References

- All source code will be available on this public repository:
    - https://github.com/Feras-dev/CS259_term_paper
- I'm happy to answer any questions via email:
    - feras.alshehri@sjsu.edu

Thank you

Feras Alshehri

feras.alshehri@sjsu.edu

https://github.com/Feras-dev/CS259_term_paper