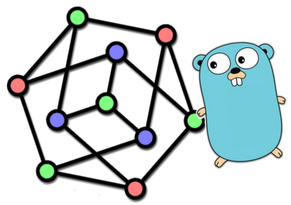
GraphMatch : Powered by Concurrency & Parallelism

McMaster University

Enamul Haque, Emil Sekerinski ~ Computing & Software

Introduction

GraphMatch provides a Sub-graph Pattern Matching library for certain types of patterns. It uses naive searching algorithm to find the patterns in the Main-graph from the Sub-Graph. And also made good use of concurrency techniques using Golang's Channels and Parallelism in the processor.



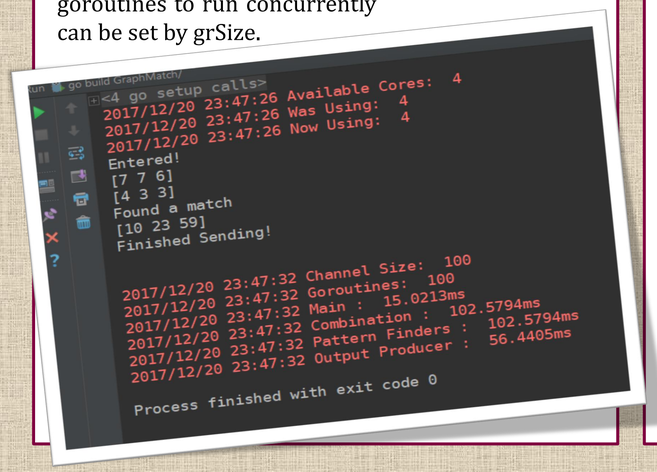
The main focus of this experiment was to utilize Google's Golang which has lightweight but faster channels for concurrency and to see how much it can be scaled for combinatorial problems.

GUI & Input

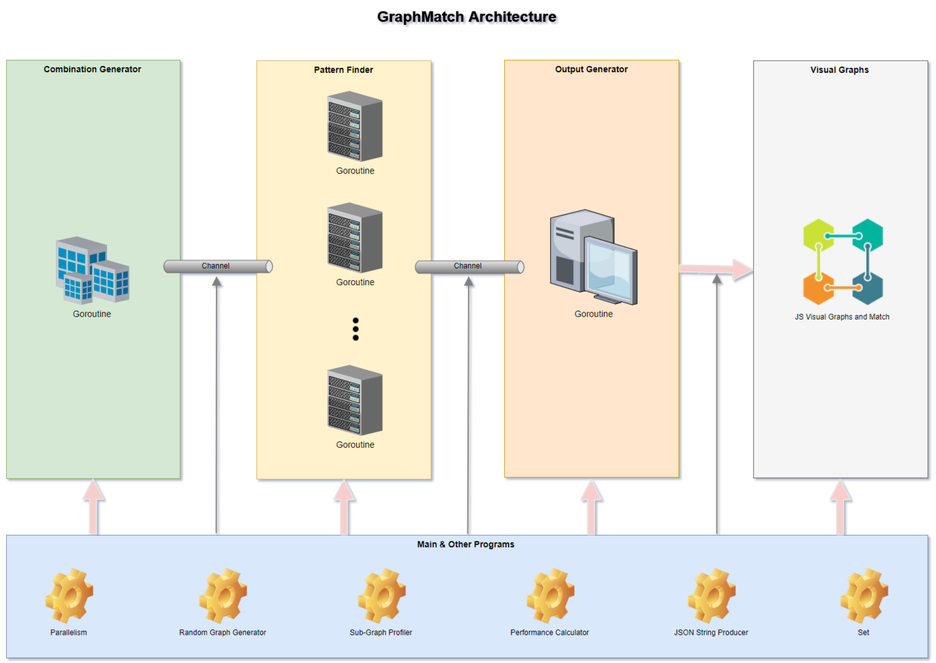
Number of Cores (1-4) as parameter to the Parallelism function. Number of nodes for the main graph can be set by gSize and for Sub-Graph by sgSize. Then the size of the buffer in the channel for Combination generation by chSize and also the number of goroutines to run concurrently can be set by grSize.

```
Parallelism(4)

var gSize = 500
var sgSize = 30
var chSize = 100
var grSize = 100
```



Architecture



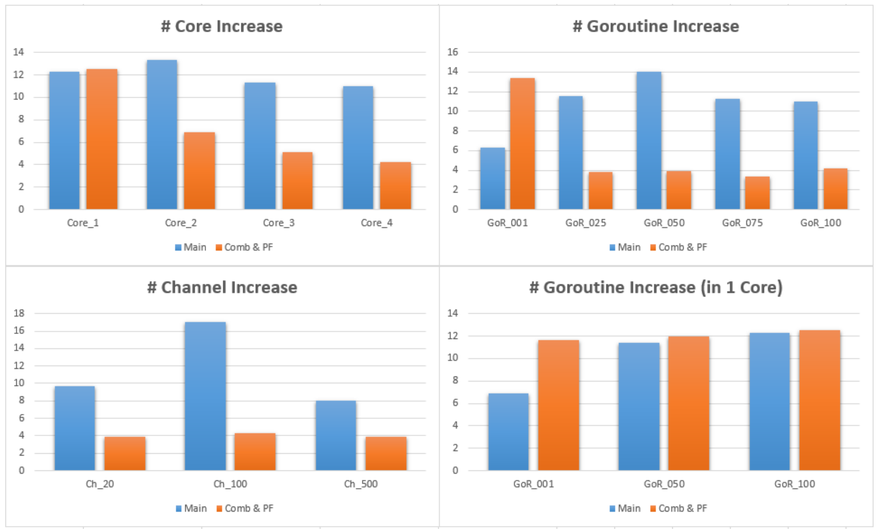
Combination Generator: Generates all the possible combinations of edges from the edge set of the graph.
Pattern Finder: Search Patterns with grSize channels concurrently.
Output Generator: JSON file creation for graph output. This module receives output graph from the goroutines concurrently.
Visual Graph: JS graph.

Besides main modules, it has sub-modules for random graph generator, sub-graph info profiler, performance calculator, JSON string producer and Set library.

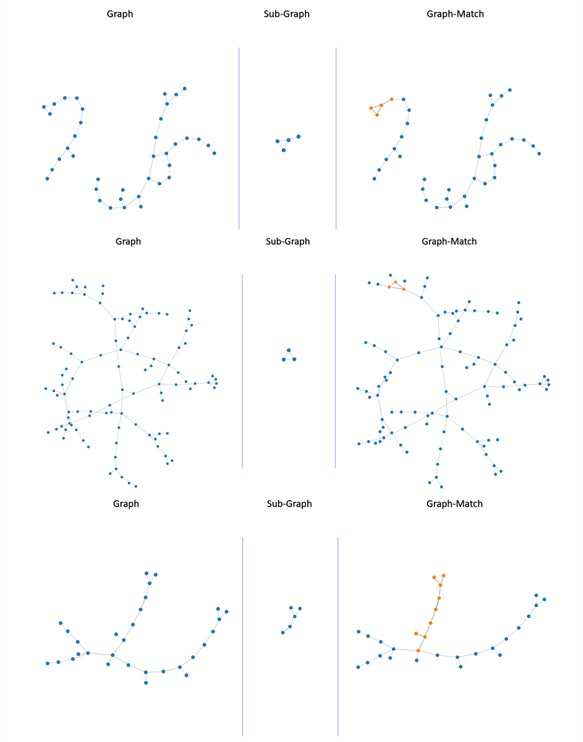
Experiment Results

- ✓ *Number of core increase speed up goroutines performance!*
- ✓ *Increasing goroutines helps up to certain numbers of them!*
- ✓ *Channel size doesn't help much!*
- ✓ *Goroutines put some pressure on the main function!*

1. Running Time of Main function (in ms)
2. Running Time of Combination & (up to the last) Pattern Finder goroutines (in s)
3. Number of Cores
4. Number of Goroutines (As Pattern Finder)
5. Size of the Channel (Combination => Pattern Finder)



Output Graphs



Future Research

- Parallel combination generator may improve the speed but the impact of 2 sets of parallel goroutines may slow it down also.
- Is there any number for concurrent goroutines which can be stated as a better choice for different kind of algorithm designs?

References

[1] Akl Selim G, Gries. D, Stojmenovic. I: An optimal parallel algorithm for generating combinations. Information Processing Letters (vol:33, Issue: 3, Nov 1989)

[2] Blog: Rob Pike "Concurrency is not parallelism"

[3] Site: Google's Golang https://golang.org/

[4] Code: github: GraphMatch