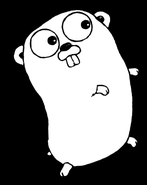
The Go Language

Google



ECE PARIS GRADUATE SCHOOL OF ENGINEERING

What is the Go Language ?

The Go language was developed by Google, it became a public open source project on November 10, 2009.

The developers realized that the existing languages did not have all the good points we can want. In fact, when you are developing, you were forced to choose between efficient compilation, efficient execution or ease of programming. Obviously, as a program, we want all of those three good points and this is why the Go Language has been created.

It supports a mixture of static and dynamic typing, and is designed to be safe and efficient. The most important question you might have is: is Go an object oriented language? There are two answers, yes and no. The Go language offers an object-oriented style of programming, however there is no type of hierarchy in this language.

Is this language really efficient ?

Presentation of our tests

To verify what Google claims, we decided to implement a test in order to compare the Go language with Java and C language.

The purpose of the test is to compare the execution time between different languages using the same algorithm. We decided to implement a classic sort of an array of integers. To have significant runtime values the size of the array is 10000.

First, we created a binary file containing the values to be sorted. This file is loaded in each program. Thus, the three arrays sorted are exactly the same, to have the best possible comparison.

Then, the array is sorted using a classic and easy to implement bubble sort (performance O(n2)).

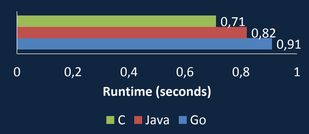
Some Go Code

For example, here is the code used in Go to sort the array :

```
var temp int = 0;

for j:=0; j<9999; j++ {

    for i:=0; i<9999; i++ {

        if(tableInt[i]>tableInt[i+1]) {
            temp = tableInt[i];
            tableInt[i] = tableInt[i+1];
            tableInt[i+1] = temp;
        }
    }
}
```

Results of the tests

The execution of the three programs gives us the following results (average of 100 executions):



The Go language is the slower with 0.91s, followed by Java with 0.82s. The faster is the C language with 0.71s.

This test is, obviously, showing the performance of some very simple operation (comparison, assignments) and doesn't take into consideration all the aspect of programming. Some benchmark have been realized with more algorithms, structures, and handle memory patterns.

What are the claimed advantages

As said in the description part, the Go language tries to make developing more simple and more efficient. For that, it regroups the three points a developer is looking for: efficient compilation, efficient execution and ease of programming.

In order to see if the Go language really achieves these goals we will do several tests in the third part. However, we can analyze the complexity of the language without doing tests and comparing it to other languages. Why is the Go language easy to program?

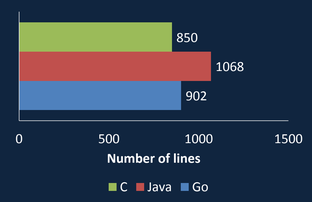
There are several reasons for that. One of them is the fact that there are no more forwarded declarations, no header files ; everything is declared only once. The syntax is clear and easy to understand. This is very important, as trying to find the keywords can be a real pain when you have hundreds if not more of lines of code.

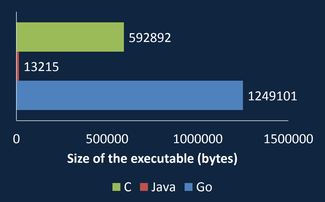
The Go language has a different approach to programming than the totally object-oriented languages. Usually, when coding with those languages we think too much on the relations between types. In Go we do not have to declare everything ahead to make everything work, Go satisfies any interface that specifies a subset of its methods. Go's approach is relations are important in the code, not the objects (classes). In other words, Go allows you to have types which contain data and methods that operate on that data and it allows you to create interfaces in which you can place those types. Interfaces are abstract representations of behavior (sets of methods). If an object implements the methods of an interface, then it has that interface as a type. The developer has nothing to do.

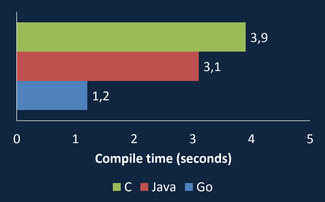
Other benchmark realized :

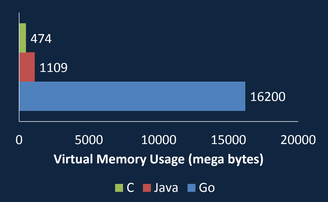
We also studied a benchmark realized in C++, Java and Go among others. The benchmark has been realized by Robert Hundt from Google. The algorithm on which the benchmark was done used : structures (lists, maps, lists and arrays of sets and lists), a few algorithms (union/find, dfs/deep recursion, and loop recognition based on Tarjan), iterations over collection types, some object oriented features, and interesting memory allocation patterns.. It allows comparison of language features, code complexity, compile time, binary sizes, run-time and memory footprint. It is important to notice that it does not cover the multi-thread, which is not optimal because the go as been designed to easily handle the multi-thread requirements which C++ and Java had not. It's a point on which Google communicate a lot but which apparently has not be tested in benchmarks.

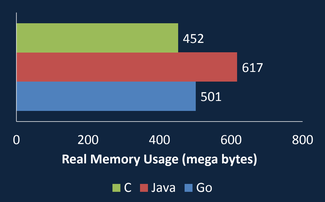
The study had two versions in each languages, the standard version which is the version made by the author himself, and the optimized version, which implements the same functions but had been corrected and improved by some external programmers. We focused on the non optimized version of these study as the optimized version can't not be done by the average programmers and because those 3 languages has not been optimized with the same effort as C++ requires more effort to be fully optimized.

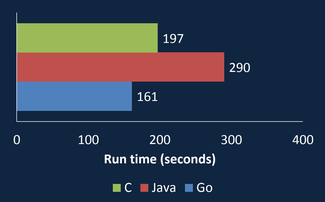










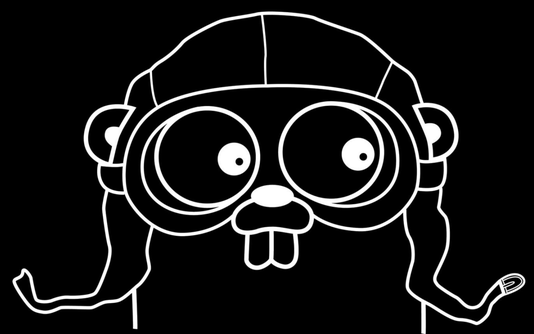


These results lead us to conclude that Go offers an interesting overall performance in terms of running time, compile time and number of lines while it suffers a lack of optimization in the binary size and the virtual memory handling. It appears that Go is not the revolution that Google pretended it to be, however, the fact that the Go compilers are still immature can lead to a lack of optimization that can reflect both on the executing time and the binary size.

Conclusion

The error margin of all this comes from the multi-thread tools of Go which has not been properly tested. If it's powerful enough and if the need is real, it could become necessary to code in a language that has been design to handle multi-threading. As actually, processors mostly progress by adding new cores rather than upgrading the single performance of cores, a complete use of those multi-thread may be necessary to take advantages of it.

Go seems to be a language with good performances and which responds to some needs in the programming world. However, it is difficult to change a standard, and even if Go could become a new widely used language in various systems, it would not be in the next few years, except if some universities begin to use it in courses which is very unlikely. Go is very young and some improvement may occur in the next few years, in terms of compilers, resources, capabilities and libraries. The real need of Go language may be for next years.



Department of Information Systems

2011/12/05

Dominique Gigon – Raphael Odienne - Laurent Wartel