Who's Julia?

Simon Knudsen, Sonni Jensen, Asbjørn Jensen

Department of Mathematics and Computer Science,
University of Southern Denmark

1. Julia

Julia is an object-oriented programming language, which has been under development since 2009. First released in 2012 and the newest stable version of Julia is version 0.4.5. The language has been created because the developers wanted a language with all the features they like from other languages.

The developers wanted the language to be open source, which means that everybody can read and modify the language. One of the ideas was to make the language as simple, readable and easy to learn as possible. The language is made for high performance and scientific computations while still supporting general purpose programming.

2. Goal

The goal of this project is to benchmark Julia and do speed comparison with Python, C++ and Java. The run time is measured as CPU time with the terminal command **time** and built-in profiling tools in the given language.

3. Syntax

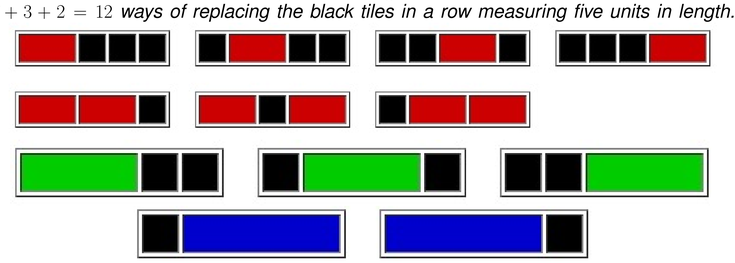
```
a = 0

for i = 1 : 10
   a += i
end

while a > 2
   i -= 1
end

function hello(name)
   println("Hello $name")
end
hello("Julia")
```

```
b = 20

for i = 10 : -1 : 1
   b -= 1
end

if a < b
   println(b / a)
elseif a > b
   println(a / b)
else
   println(a + b)
end
```

```
array = fill(1, 100)

for i = 2 : 100
   array[i] += array[i-1]
end

for i in array
   println(i)
end

minimum(array)
maximum(array)
sum(array)
```

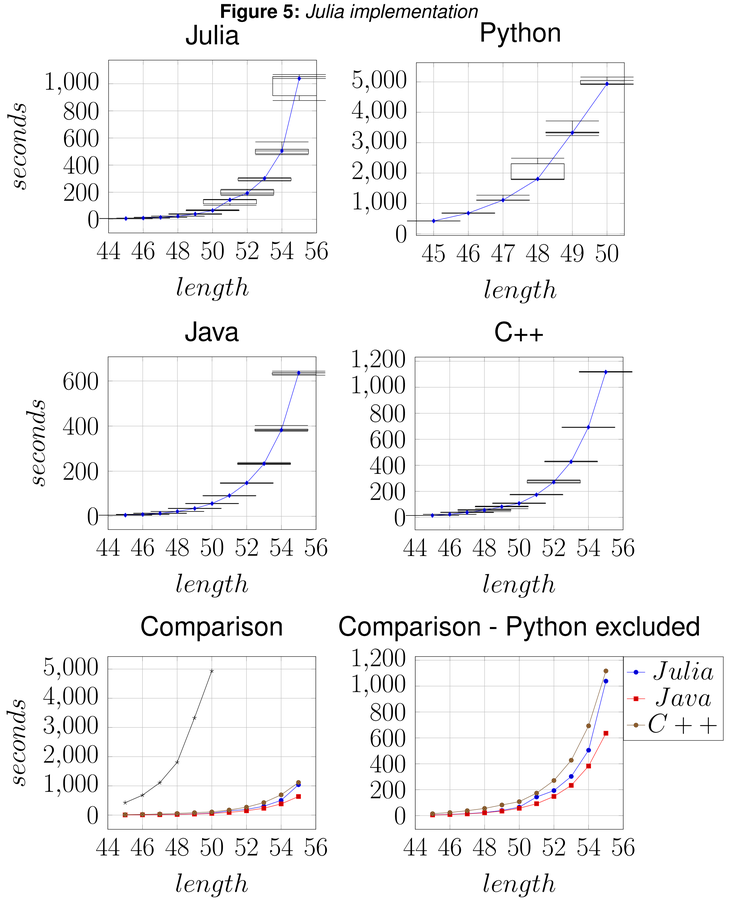
4. Project Euler problem 116

A row of five black square tiles is to have a number of its tiles replaced with coloured tiles from red(length two), green(length three), or blue(length four). If red tiles are chosen there are seven ways. If green tiles are chosen there are three ways. And if blue tiles are chosen there are two ways. Assuming that colours cannot be mixed there are $7 + 3 + 2 = 12$ *ways of replacing the black tiles in a row measuring five units in length.*



```
1  function solve(tileSize, length)
2     if tileSize > length
3        return 0
4     end
5     solutions = 0
6
7     for i = tileSize : length
8        solutions += 1
9        solutions += solve(tileSize, length-i)
10    end
11
12    return solutions
13 end
14 %
15 function calc(length)
16    result = solve(2, length) #Red tiles
17    result += solve(3, length) #Green tiles
18    result += solve(4, length) #Blue tiles
19
20    return result
21 end
```

Figure 5: *Julia implementation*



4.1 Results of Project Euler 116

It is clear that the running time is increasing exponentially. Another thing to notice is that the length of input is increasing by one, but is making a huge difference in the run time. One of the reasons is that the problem is solved with recursion, for every extra one bit of space added to the black box a lot more recursive calls will have to be made.

The differences in time between the four languages are as expected. Python does not support any form of tail recursion optimization and the result is really slow. Java and Julia on the other hand do a good job at optimizing the recursive calls and is actually faster than C++ - keep in mind that no compiler flags were used in C++, so the default optimization level is used.

Julia and Java are close, but Java is a bit faster, which is expected because of the fact that Java has been in development much longer than Julia and has a lot more optimization than Julia.

5. Pros and Cons of Julia

Advantages:

- Julia is easy to learn and software are easily made.
- Julia does good compiler optimizations, which sometimes makes Julia as fast as C++.
- Overall, Julia performed in the tests very well.
- Julia was the only language that did not cause any out of memory issue at some point.

Disadvantages:

- Compiler optimization can break (Dual pivot quicksort)
- Missing features and weird behaviors with implementing code from other languages.

6. Conclusion

It can be difficult to make any conclusions by the benchmarking, but it may give an idea of how Julia performs, compared to Java, Python and C++. One thing to keep in mind is that Julia is in version 0.4.5 - it has yet to reach version 1.0.0. This could mean that Julia might get even faster in the future and by the benchmarking it does look as Julia is already performing well.

The developers are trying to make a programming language that is as easy to learn and write as Python, but with a compiler that optimizes code like Java and both of these goals have more or less been reached already. Julia was definitely easy to learn and by the benchmarking it does look like that the compiler is doing well with the optimization. It also feels like the Julia developers are playing with the idea of giving the user some control over what is happening behind the scene. 'Playing' because this does not feel close to done. For example, the user can allocate memory for a variable manually, but if a new value is assigned to the variable, the memory size is also changed. It is not possible to free a declared variables memory entirely and the only operation the user can do with the garbage collector is force garbage collection and disable the garbage collector.

References

Julia's offical homepage
www.julialang.org

Project Euler
www.projecteuler.net