A comparison of the concurrency features available in the Go and Clojure programming languages

Stephen Adams
University of Minnesota Morris
600 E 4th St.
Morris, Minnesota USA
adams601@morris.umn.edu

ABSTRACT

In a 2005 interview Gordon Moore predicted that his namesake law (Moore's law states that the "number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years") will reach a fundamental limit in ten to twenty years [5]. Our transistors are starting to approach the size of individual atoms which is a boundary that will not be broken. With this in mind, the challenge to continue increase the speed of computing will soon fall to computer scientists as our abilities to create programs that utilize multiple processing cores will be where the futures speed gains are created.

In the past five years two languages were created (Go and Clojure)[6] with support for easy concurrency being a primary tenet of these languages. Both of these languages have received significant levels of interest (both languages are in the top 100 programming languages in the Tiobe index [2]) to think that they may either see large scale adoption for their concurrency features or influence the next generation of concurrency control methods. In this paper I will briefly cover the basics of both Go and Clojure syntax and then I will cover these two languages' concurrency features. Finally there will be a discussion about the merits of both of these concurrency systems.

Keywords

Concurrency, Programming Languages, Software Transactional Memory

1. INTRODUCTION

"A big challenge for concurrent programs is managing mutable state. If mutable state can be accessed concurrently, then as a programmer you must be careful to protect that access."[1] This quote by Stuart Halloway succinctly states why the traditional model of concurrency control is so difficult to manage properly, and consequently why it's seldom implemented. Clojure and Go both provide alternative, and different, concurrency control. Clojure through its software transactional memory[3] and Go provides goroutines which are a lightweight thread[4].

This paper will first provide a brief overview of Clojure then the Go programming languages. Then specifically the concurrency features of these languages will be covered more in depth. Finally a comparison of the two concurrency methods and a brief conclusion.

2. LANGUAGE BASICS

In this section the very basics of Clojure and Go will be explained. The idea here is to provide the barest minimum of syntax knowledge so that code examples can be understood. If this section is not adequate or the reader is further interested in one of these languages `http://clojure.org` or `http://golang.org/` should provide plenty of information concerning their respective languages.

2.0.1 An Introduction to Clojure

The first thing that becomes evident when looking at Clojure code is that it is a Lisps and therefore uses prefix notation like so:

```
(+ 2 3)
```

Which returns the value 5.

3. CONCLUSION

4. REFERENCES

[1] R. Hickey. *Programming Clojure*. The Pragmatic Programmers LLC, 2009.
[2] T. Software. Tiobe programming community index for april 2012, 2012.
[3] Wikipedia. Clojure, 2012. [Online; accessed 19-April-2012].
[4] Wikipedia. Go (programming language), 2012. [Online; accessed 19-April-2012].
[5] Wikipedia. Moore's law — wikipedia, the free encyclopedia, 2012. [Online; accessed 16-April-2012].
[6] Wikipedia. Timeline of programming languages — wikipedia, the free encyclopedia, 2012. [Online; accessed 16-April-2012].