Containerization Support Languages

Haonan Zhou – University of California, Los Angeles

Abstract – The goal of this report is to explore an alternative language in order to implement a Linux container called Docker. Java, Python, and Groovy are three languages discussed and compared with Go in Docker. The conclusion is that Groovy is the most suitable replacement for Go in order to implement Docker.

Introduction

The motivation of Docker comes from the desire to run a program on virtual machines. The problem with tradition virtual machines is that it has too much overhead. Even for a really simple program, a large amount of time can be wasted just waiting for the VM to start. Docker, as a Linux Container (LXC), provides an alternative solution. An LXC offers almost the same virtualization as a VM but requires little overhead. For purpose of the proxy herd program from the project, Docker is a great replacement of VM since it enables the program to run under the same Linux environment with a Linux kernel.

Go

In the talk by the Docker Team, five reasons are given on why they chose Go to implement Docker:

1. Static Compilation –
   a. The team claims that static compilation of Go reduces dependencies and makes installation easier. This is one of the most important merits of static linking. If there is any bug in the code, it is immediately caught at compile time. At run time, the library modules are already loaded into the binary executable so that it is guaranteed that there are no bugs related to linking or using library.
   b. An important advantage of dynamic compilation is that size of the executable will be greatly reduced. With dynamic linking, the compiled program is smaller in size. This can be a real issue if the code requires a lot of libraries. However, Docker must bundle the library files anyway, even if it uses a dynamic linking language, since most users probably do not have the required libraries installed on their machines. Therefore, the Docker team makes a valid argument in choosing static linking.
2. Neutral –
   a. The Docker group also identifies Go as a "neutral." This a rather vague and confusing assertion and I hardly find it supportive. Since Go is a language invented by Google, I do not see why it is considered a "neutral" language.
3. It has what is needed
4. Full Development Environment
5. Multi-architecture build
   a. The three reasons above can be discussed together. Each of them are valid statements, but each of them have a common weak point.

That is, Go is not the only language that has these features. For example, the talk mentions that Go has extensive libraries. This is a feature also shared by Python and Ruby. In fact, the Python community is much larger than Go, so there are probably a greater number of standard libraries in Python. The team also mentions that Go is ducking typing. Similarly, Python and Ruby both has duck typing features, which does not explain why Go is preferred in this case.

Java

The most noticeable drawback of using Java in DockerAlt would be that it would be very hard to use duck typing with Java. There is, indeed, a reflection module in Java that allows programmers to simulate duck typing. However, since Java is not designed to use duck typing, it is unrealistic to use this module extensively in code that heavily relies on it. After all, if there are other languages that enables duck typing easily, we should switch to that language.

Another arguable disadvantage of Java, compared to Go, is that the language itself is hard to learn. Since Go is a scripting language, a beginner can learn Go very easily. For Java, it may take someone years to become an expert in the language. Also, the readability of Java is considerably lower than Go. It can be easy to train someone to understand a Go program, whereas a simily Java program can be long in size and much harder to read. If we want the DockerAlt to be easily understood and maintained, Java is probably not the best option.

Python

As discussed earlier, Python provides at least the same number of libraries as Go. There should be no trouble to rewrite the same Docker implementation in Python. Also, like Go, Python's portability allows DockerAlt to run on multiple architectures with no problem.

The main difference of Python is that code is interpreted rather than compiled. This means that all library modules are loaded at run time. The program will run slower due to the library loads. Also, there is a risk that a new released library may not be compatible with the code. In a sense, the developer has less work to do, but the users would have more trouble. Generally, this is the situation we want to avoid.

Groovy

Groovy shares a lot of features with Go. First of all, it uses static compilation, which, as discussed above, yields better performance and reliability than an interpreted language such as Python. In addition, Groovy uses duck typing by default, a characteristic desired by Docker developers. It also has libraries exploiting system concurrency and parallelism, providing the development environment mentioned by the Docker team.

Since Groovy is derived from Java, a Java programmer would find no trouble adapting it. And since it is a scripting language, a programmer would find it easy to learn even if he or she is not a Java expert.

Conclusion

After comparing and contrasting the four languages, it seems to me that Groovy is the best replacement for Go in DockerAlt.

Sources:

http://groovy-lang.org/index.html

http://cs-fundamentals.com/tech-interview/c/difference-between-static-and-dynamic-linking.php

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html