Containerization support languages

Chengyu Wang
University of California, Los Angeles

Abstract

Docker is a new software technology that is based on Linux servers. It is implemented in Go language. In this paper, we will explore the possibility of implementing Docker with the following three candidate languages: Ocaml, Java and Scala. This paper will discuss each language's pros and cons and finds out the most suitable one.

1. Introduction

Docker is a technology that is used to replace the role of virtual machines that could provide an isolated environment to run applications and tests. Containerization is the process of packaging dependencies, namespaces and other specific application libraries into a single container as an image. Docker packages applications and their dependencies together into an isolated container making them portable to any infrastructure and thus eliminates the "works on my machine" problem.

Unlike virtual machines, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application. Also, Docker is more space-efficient than virtual machines. These properties make Docker easier to develop and maintain.

2. Go programming language

Go is a language developed by Google. The following reasons make it a fit to implement Docker.

The first reason is that Go has static compilation. This property helps Go reduce dependencies. Once a bug is found, programmer does not need to check linking libraries since all library modules needed have already been loaded local. Thus Go is easier to install, test and adopt.

The second reason is that Go is neutral which means that Go combines the advantages of many mature and popular languages. Since it's a relatively new language, it does not have any serious stigma.

The third reason is that Go has many attractive properties. Go has good asynchronous primitives, which means that the program can wait for I/O while continuing the other work. Go also has many low-level interfaces which help manage process calls and system level calls. Besides, Go has extensive standard library and data types for programmers. Its duck typing mechanism make it easy for programmers to write simpler code without worrying too much about types.

The fourth reason is that Go addresses multiples issues of development workflow. The programmer can see documentation for any package, fetch dependencies, solve "tab" and "space" problem once for all time and run tests and prototypes easily with just a simple command.

The last reason is that Go has a multi-arch build that does not require pre-processors, a boost of performance.

At this point, Go seems very promising. However, Go also has some drawbacks that cannot be ignored. It does not have thread-safe Maps, which is a tradeoff for fast performance. Programmers need to ensure the data safety by themselves. Another obvious possible drawback is that Go is not an object-oriented programming language. It has no support for generics and might cause a lot of repetitive code in the program.

3. Java

Java is a very popular object-oriented language. It is a very mature language with libraries that can nearly satisfy everything a programmer will need.

Java uses static type checking, which is similar to Go. Although Go's many properties (such as duck typing) make it like dynamic type checking, it's essentially

static type checking. Static type checking makes Java report errors before running and much safer. It is a plus for Java.

Another biggest advantage of Java is that Java is complied to immediate bytecodes by Java Virtual Machine so that Java code is very portable. It can run on any hardware that installs JVM.

Java's disadvantages for implementing DockAlt are also very obvious. Java is much more complicated than Go in some ways. Duck typing of Go makes coding of Go much more intuitive and easier than Java. Java's complicated coding requirements make it not a good choice for fast prototyping and thus make the development and testing cycle longer.

Java's multi thread working environment is also a very important concern. Programmer needs to ensure the data safety among threads, which requires a lot of extra code.

Another important reason for not choosing Java is that Java lacks Linux Container API. Programmers will need third-party libraries, which might introduce some unexpected bugs and increase the coding difficulty.

Overall, Java has some advantages but its disadvantages make it not a perfect fit for implementing DockAlt.

4. Ocaml

Ocaml is a well-known functional programming language that is really different from both Go and Java from the perspective of coding style.

It's similar to Go in that Ocaml also has static type checking and duck typing. This property ensures a safe environment of implementing DockAlt and a rapid prototype since the programmer does not need to explicitly specify the type.

Another advantage of Ocaml is that Ocaml is free of side effects. While implementing the DockAlt, we do not need to worry about data safety or race conditions. This is a valuable property especially for large applications.

Its disadvantages are very obvious. The biggest disadvantage is that Ocaml is a functional programming language that is really hard for beginner to master in a short time. This feature also makes Ocaml hard to read.

Another disadvantage is that Ocaml is not a very popular language compared to Java. It does not have extensive libraries for programmers. It's harder for programmer to build such a big application as DockAlt from piece by piece without external library support. Also, similar to Java, Ocaml lacks Linux Container API.

Overall, Ocaml is not a very suitable candidate for implementing DockAlt due to its functional programming nature and lack of libraries support.

5. Scala

Scala is a relatively new general-purpose programming language that provides support for functional programming language. In some sense, we can say Scala is a combination of object-oriented language and functional programming language.

Similar to Java, Scala is compiled to bytecodes by Java Virtual Machine so that Scala has perfect portability. Programmers only need one DockAlt implementation for all machines who has JVM. Also, Scala has static type checking that reports errors at compile time. Thus a safer development environment is ensured.

Another advantage of Scala is that Scala can use Java's libraries. Thus, although Scala is a relatively new language, it has extensive library support from Java.

Also, Ocaml's functional programming property makes the programmer easier to write code that does not need to worry about safety and race conditions in multi threaded environment. This is a very valuable property since it combines the advantages of both Java and Ocaml with data safety and fast performance.

Its disadvantages also come with its benefits. It's different from traditional object-oriented style programming language like Java or C, which makes it hard to learn for beginners.

Since Scala is relatively new, its community is also small compared to Java. Sometimes when programmer meet some trouble, they cannot find immediate support from online due to small community size. If developers do not have the time to resolve issues themselves, Scala may not be a viable option for them.

Scala also has terrible error tracking mechanism. In Java, the compiler will give complete tracebacks when error is reported. However, in Scala the programmer

needs to do this job by themselves. For a large application like DockAlt, the debuggability is really needed. This property makes Scala less attractive.

Overall, Scala has good performance and portability but requires the programmer to be really high-level to solves the issues by themselves since the community is relatively small.

6. Conclusion

After reviewing Go, Java, Ocaml and Scala, I believe Go is still the best option for implementing the Docker application. Its built-in Linux Container API makes it very convenient for developers to use. Among the other three options, I will say perhaps Scala is the best. Java's multi-thread mode makes its data vulnerable to be changed and Ocaml's lack of library support makes it hard for developers to build application easily. As for Scala, although it's a new language and is still growing, its combination of performance and data safety makes it a better candidate. Experienced programmers in Scala can possibly build DockAlt with the least effort compared to Java and Ocaml.

Reference

[1] https://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/24-Why_Go5_multiarch_buildwithout_preprocessorslinux godarwingo

[2] https://www.docker.com/what-docker#/overview

[3] https://opensource.com/resources/what-docker

[4] https://www.scala-lang.org

[5] https://softwareengineering.stackexchange.com/questions/77436/is-googles-go-a-type-safe-language

[6] https://andrumyers.wordpress.com/2016/06/16/java-vs-ocaml-vs-scala/

[7] https://www.toptal.com/scala/why-should-i-learn-scala