Homework 6. Containerization support languages

Alvin Lim, 804011675
University of California, Los Angeles

Abstract

In this paper we first explore Docker, an open source platform that provides lightweight, portable, containers for packaging, shipping, and running applications. Docker is implemented in Go, which utilizes Linux containers for building lightweight environments for easily developing and executing programs without dependency issues. We will also study the feasibility of using Java, Python, and Scala to develop a potential platform, DockAlt, which would achieve the same functions as Docker.

1. Introduction

Docker automates and encapsulates software deployment in lightweight, portable, self-sufficient containers with the Linux Container (LXC). The platform could be thought of as a middle ground between virtual machines and static linked binaries [1]. While virtual machines emulate an entire machine using virtualization, this results in large overhead and therefore limits how many virtual machines one can concurrently run on a physical machine. Static linked binaries require an environment where everything is provided to execute the application, which is not always the case.

1.1. Go

Modern programs are required to scale and become more complex, making software management and deployment much more difficult. A popular approach this problem is utilizing virtual machines, since one and distribute virtual disks that contain the development software with the dependencies and environment it needs.

Docker tackles the issue in a similar fashion, using a more lightweight approach of LXC, which provides operating system level virtualization to run multiple encapsulated Linux systems on a host [2]. LXC allows Docker to sandbox processes into namespaces and network interfaces [3]. This allows developers and ops to develop and deploy the application separately.

Before choosing to use Go for implementing Docker, the developers had used Python for a previous project, dotCloud. For Docker, however, they wanted to start from a clean state and use a more neutral language to appeal to a wider range of customers and platforms. In addition, Go provided several features which are desirable for the Docker platform: C-like syntax, duck typing, static compilation, built-in concurrency primitives, extensive libraries/datatypes, and fast compilation due to lightweight type analysis [4][5]. While Go is a relatively new language with bugs to work out, it was chosen to implement Docker because it provides a good standard library, package management system, and is easy to read/work with.

2. DockAlt

2.1 Java

Due to its open source nature, Docker greatly depends on contributions from the open source community – the ease of learning Go makes it easy for the community to contribute. Java is a strong and static typed language, meaning it

has reliable performance. It is also platform agnostic, like Go.

The problem with using Java for DockAlt is that compared to Go, it is a complex language with syntax that is comparatively verbose, which makes it more difficult to understand and use. Additionally, Java doesn't have an LXC API, which would pose a problem in implementing DockAlt. LXC interfaces could be implemented by third party libraries, but this would cause problems in development. Another alternative would be to use the Java Native Interface (JNI), which calls native libraries and code from other languages, but this would rule out Java's advantage of being a platform agnostic language.

2.2. Python

As a dynamically typed language, Python provides duck typing like Go, allowing for greater flexibility in development. It also has a LXC binding, which would allow DockAlt to continue to use the same containerization method as Docker.

Python's syntax is simple to read and use as well, making it viable to implement Docker in. While Python is quite similar to Go in the aspects of simplicity and flexibility, the Python interpreter is relatively slow, since Python is read and compiled at runtime [6], so for DockAlt's objective of running various applications on a lightweight platform, this may not be as efficient.

2.3 Scala

Scala is a language that advertises having elements of object oriented programming and functional programming [7]. Like Go, Scala is also a platform agnostic language, running byte code in JVM. Since Scala runs on JVM, it has relatively good safety and reliability due to its static type checking which is better compared to runtime typechecking when working with larger software. This also means that Scala is highly compatible with Java, making many of Java's extensive collection of packages usable as well.

The problem with Scala is that it is largely functional, making it more difficult to work with for users who are not familiar with functional programming. It is also not as popular as Java and Python, so has a smaller community – this would not work well for DockAlt, since we would like a large community to contribute to the open source platform or use an easy and familiar language that newcomers can pick up.

3. Conclusion

We considered three alternative languages with which to implement DockAlt: Java, Python, and Scala. Out of these languages, we find that Python is the best choice for implementing DockAlt. It is very similar to Go in many ways, such as being easy to learn/use and allowing for duck typing. These similarities make it easy to see why the Docker developers chose to use Python to implement dotCloud. Python also has support for LXC binding as well, which is very useful since the implementation of Docker uses it.

References

[1] "Docker." Docker. Web. 07 Mar. 2016. <https://www.docker.com/>.

[2] "Docker." Github. Web. 07 Mar. 2016. <https://github.com/docker/docker>

[3] "Linux Containers." LinuxContainers.org. Web. 08 Mar. 2016. < https://linuxcontainers.org/>

[4] "Go." Wikipedia. Web. 08 Mar. 2016. <https://en.wikipedia.org/wiki/Go_(programming_language)>

[5] "Go." Go. Web. 08 Mar. 2016. <https://golang.org/>

[6] "Python." Wikipedia. Web. 09 Mar. 2016. <https://en.wikipedia.org/wiki/Python_(programming_language)>

[7] "Scala." Scala. Web. 10 Mar. 2016. <http://www.scala-lang.org/>