DockAlt: An Alternative to Docker

Ryan Trihernawan
University of California, Los Angeles
June 3, 2015

Abstract

DockAlt is alternative implementation of Docker written in a different language. This paper discusses the technologies' effects on ease of use, flexibility, generality, performance, and reliability. Moreover, it explores the most-important technical challenges in rewriting Docker in three different languages-Java, Python, and Julia.

1. Introduction

LXC is an operating-system-level virtualization environment designed for running multiple isolated Linux containers or systems on a single Linux control host. It provides many benefits that a virtual machine (VM) offers without the overhead due to running a separate kernel and simulating all the hardware. Such advantage is plausible because LXC runs as an isolated process in the user space on the host operating system (OS), sharing the kernel with other containers and, where appropriate, binaries or libraries. In short, the most important differentiating factor between VMs and containers is that while hypervisors in VMs can use multiple different OSs, containers must share the same OS.

Docker is a runtime for Linux containers written in Go programming language. Since its initial release on March 13, 2013, it has gained widespread popularity and use in major tech companies such as Ebay and Baidu to name a few. Containers have been around since year 2000, but they did not perform as well as competing technologies. Docker offers easier and safer deployment of containers compared to its predecessors. It also helps initiate standardization to containers by partnering with companies in the same space such as Google, Red Hat, Parallels, and Canonical. Most importantly, it allows developers to pack, ship, and run any application as a LXC container that can run virtually in any system. Its portability lets developers to collaborate with others to develop, test, and ship their application regardless of their host environments.

2. Go

Go is the chosen language for Docker. It was initially conceived in Google about 6 years ago. Since its initial release, it has attracted many enthusiasts since it is easy to learn. It has a static type and loosely adopted C syntax with some dynamic-typing capabilities; hence it provides a strong duck typing. Besides these typing capabilities, it is also easier to install, test, and adapt compared to popular languages such as C, C++, Java, and Python, thus it is a good candidate for bootstrap. These features motivated Docker developers to use Go. Moreover, it offers what these developers want such as good asynchronous primitives, low-level interfaces, and extensive standard library and data types. Lastly, its full development environment and multi-arch build capabilities help accelerate the development process.

3. Java

3.1. Overview

Both programming enthusiasts and large corporations have used Java, one of the most popular languages, in many different applications. Instead of providing a multi-architecture build, Java allows developers to run a compiled Java code that is runnable on all platforms with Java support without the need for recompilation. This capability is achieved by having a Java code compiled to bytecode that is runnable on Java virtual machine in any computer architecture. This portability suits the need of DockAlt to be portable. It also has extensive standard library and data types. It offers low-level threading and asynchronous programming capabilities. For instance, Java's Concurrency Utilities framework supports asynchronous programming suitable for DockAlt. Unlike Go, it has an elegant exception handling and thread-safe maps. These two features will eliminate the undesirable Go code to handle both situations. There exists an integrated development environment (IDE) written in Java. However, its C-inspired syntax means that it is technically more challenging for beginners to learn. Programmers with experiences in C

and C++ will find Java's syntax easy to understand. Lastly, like Go, Java implements garbage collection.

3.2. Binding to LXC

Java does not provide an official binding to LXC. However, there exists a third-party library that provides such binding. lxc-java is a Java API for LXC available on GitHub. A third-party library is not subject to robust reliability and compatibility control like a standard library is, and thus careful examination of the library is needed. An alternative is to use Java Native Interface (JNI), which allows for a two-way communication (between caller and callee) with libraries written in other languages such as C and C++. Thus the developers can write a Java code that calls LXC API in C. This level of abstraction introduces another complexity in the development process.

4. Python

4.1. Overview

Another mature and proven language, Python has a large user base. It has the most similarities to Go among all the three candidate languages. Firstly, it supports a strong duck typing, thus making it easy to learn for beginners. Secondly, it is used in dotCloud, a predecessor of Docker by the same company. dotCloud developers built Docker to improve some of dotCloud's features such as binding and APIs, and to introduce some new concepts. Since dotCloud developers build Docker, they have proven that Python is suitable for this kind of application (Platform as a Service or PaaS in short). Python provides many of the features that Docker developers need such as good asynchronous primitives, and extensive standard library and data types. For instance, asyncio package is included in the standard library on a provisional basis. Moreover, there exist third-party libraries such as Twisted that further simplifies asynchronous programming. Like Go, Python also implements garbage collection. Unlike Go, there exist Python-specific IDEs such as PyCharm, Wing IDE, and PyDev. These IDEs allow for easier development.

4.2. Binding to LXC

Python supports binding to LXC with the lxc package, which allows for direct and easy implementation of DockAlt that uses LXC as a foundation. Moreover, dotCloud implementation shows that Python provides a convenient binding to LXC.

5. Julia

5.1. Overview

The last candidate for DockAlt is Julia. It is the newest out of all the three languages considered. Like Go and Python, it implements a strong duck typing, hence it is easier to learn for beginners. Moreover, it is a neutral language like Go in that it is not among the most popular languages such as C++, Python, Ruby, or Java. However, it has a strong relationship with C and Python in that it allows call to Python functions and direct call to C functions without wrappers or special APIs. The latter capability allows it to have a good performance, comparable to that of statically compiled languages like C. Furthermore, it offers low-level interfaces with powerful shell-like capabilities for managing other processes and thus developers have more control of concurrency in their programs. Lastly, it implements a garbage collection like Go, Java, and Python. All of the above features give Julia a strong candidacy for implementing DockAlt, however, since it is only three years old, it is a relatively new language and thus it is not as reliable as the other three languages.

5.2. Binding to LXC

There is no Julia binding to LXC, thus developers have to directly use the LXC API in C. Julia's compatibility with C allows for convenient usage of the LXC API. However, this capability requires the developers to be proficient in C because C has a different syntax and does not support duck typing.

6. Conclusion

Java, Python, and Julia are all good candidates for implementing DockAlt. Each language has its strong points that make them desirable for DockAlt implementation. Java is a proven and mature language with a strong and large community. Its only disadvantages compared to Go and the other languages are lack of duck typing and lack of direct API for LXC. Python is the strongest candidate among all mainly due to its proven use in dotCloud, the predecessor of Docker. Despite its strong relationship with applications like Docker, it has a strong and large user base and thus is not neutral language in a way that proponents of other

popular languages like C, C++, and Java are less likely to adopt Python due to their bias towards the language. For example, some C developers are less likely to adopt Python because they like the performance that C programs give compared to Python programs. Lastly, Julia is the weakest candidate for implementing DockAlt largely due to its novelty (it is only three years old) and reliability. New languages tend to have more bugs and and thus they are not reliable for enterprise applications. Julia's dependence on C to do binding to LXC puts beginning developers at disadvantage since C is one of the most technically challenging languages to learn the first time. Ignoring developers' bias towards Python, it is the winner for implementing DockAlt.

References

[1] Ben Kepes, *dotCloud, The Orphan Child of Docker, Finds A New Home*, Forbes, 2014, http://www.forbes.com/sites/benkepes/2014/08/04/dotcloud-the-orphan-child-of-docker-finds-a-new-home/, (Accessed: 2 July 2015)

[2] *Calling Java Methods Asynchronous Using Spring*, Level Up Lunch, 2015, http://www.leveluplunch.com/java/tutorials/026-asynchronous-native-java-method-calls-spring/, (Accessed: 2 July 2015)

[3] Klint Finley, *Out in the Open: Man Creates One Programming Language to Rule Them All*, Wired, 2014, http://www.wired.com/2014/02/julia/, (Accessed: 2 July 2015)

[4] Mr Spark, *Which Python IDE is best? Choose your own!*, LinuxCandy, 2012, http://www.linuxcandy.com/2012/07/which-python-ide-is-best-choose-your-own.html, (Accessed: 2 July 2015)

[5] Steven J. Vaughan-Nichols, *What is Docker and why is it so darn popular?*, ZDNet, 2014, http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/, (Accessed: 2 July 2015)

[6] Yonas Beshawred, *How Docker Was Born*, stackshare, 2014, http://stackshare.io/posts/how-docker-was-born, (Accessed: 2 July 2015)