Containerization Support Languages

Patrick Killian Jackson
University of California, Los Angeles

CS 131
March 11, 2016

Abstract – The Docker platform is a lightweight tool that efficiently packages an application, including all of its dependencies, in Linux containers so that the application is portable and runs the same in any environment. This paper examines the decision in using the Go programming language to implement Docker and the viability of Java, Python, and Rust as potential candidates to implementing an alternative to Docker, DockAlt.

1 Introduction

Docker is a platform that allows for isolation of environments, using Linux containers, that provides similar benefits to those of virtual machines. Like virtual machines, Docker containers isolate resources and allocate benefits in similar ways; however, Docker containers use a different architectural approach making them more portable and efficient than virtual machines. These containers contain an application and its corresponding dependencies, including libraries and namespaces, but each container shares the kernel with the host and all other containers on the machine. This reduces overhead by making much more efficient use of memory as entire operating systems do not need to be stored for each container, and the containers are portable as they are not tied to a particular machine. Docker allows programmers to not worry about establishing and maintaining new environments or installing tools a program might depends, instead focusing on developing features, testing programs, and efficiently shipping applications in Linux containers.

While Docker is a powerful platform, there are still issues that come with it. As previously stated, Docker is written in a relatively new language called Go, which has bugs of its own. In this paper we will discuss the decision of choosing Go as the language for Docker and the advantages and disadvantages of implementing this platform in Python, Java, and Rust in order to determine the best programming language for implementing DockerAlt.

2 Go

The Go programming language is a relatively new language developed by Google. Go was built to be "expressive, concise, clean, and efficient" [1] and is especially useful with regard to systems programming.

2.1 Advantages to using Go

The Docker team decided on Go as the programming language with which to develop the Docker platform for a variety of reasons. For one, Go programs benefit from static compilation. This feature of Go makes it easier to install, test, and adopt applications. Commands such as "go build" will embed everything a program needs to function saving the programmer from having to install the necessary packages needed to run a program. In addition, the Go language includes several specific features that are particularly useful for the Docker platform specificially. Go has good asynchronous primitives, low-level interfaces, an extensive standard library coupled with powerful data types and strong duck typing. Go also has a strong development environment that assists with several aspects of the development process. This development environment allows the programmer to easily access documentation for packages, fetch dependencies, test code, and take advantage of efficient prototyping. Specifically related to Docker, Go has the built-in libcontainer package that provides tools for "creating containers with namespaces, cgroups, capabilities, and filesystem access controls" [2].

2.2 Disadvantages to using Go

While Go excels in many areas, there are also certain drawbacks to this language selection. Many of these issues can be attributed to Go being a newer language that suffers from some bugs in the languages packages. For one, maps in Go are not thread-safe; however, this is a deliberate decision by the authors aimed to make maps fast. Despite the performance increase, it is left to the programmer to ensure the thread-safety of maps. Additionally, there are some issues related to building Go programs as problems may arise when programs share code. Each program in Go must be in its own package. Finally Go does not have an IDE and deciphering error messages can be challenging.

3 Python

With respect to DockAlt, Python is an interesting case as the first version of Docker was actually written in Python. This demonstrates that Python is a proven alternative to Go as an option for implementing DockAlt. Pyton is a powerful, widely used language that is dynamic, interpreted, and incredibly readable.

3.1 Advantages to using Python

The Python programming language is known for its concise, readable syntax and its multi-paradigm nature, meaning it supports object-oriented, procedural, and functional programming styles. Like Go, Python benefits from strong duck typing making the development of prototypes much easier. Additionally, Python has built-in Linux container binding. This is incredibly beneficial with regard to Docker since the platform is based on Linux containers. This Linux container binding includes concise function calls such as container.create(), container.start(), and container.destroy(), to name a few, that make working with Linux containers straightforward and clear. Python also is a fundamentally powerful language to code in due to its extensive package library and features.

3.2 Disadvantages to using Python

Despite the many advantages to using Python to implement DockAlt, there are also drawbacks. While Python is a generally fast language, because it is interpreted, it is significantly slower than languages such as C/C++, Java, and even Go. Due to its dynamic typing and lack of compile time error checking, working with Python requires the programmer to more thoroughly test programs and deal with the fact that many errors will only appear at run time. Additionally, while its reference counting memory management system is sufficient in some applications, its method of garbage collection has the potential to create disastrous memory leaks.

4 Java

The Java programming language is an object-oriented language based on classes and is one of the most popular languages today, especially with regard to developing web applications. It was intentionally created to posses very few dependencies allowing it to be portable to many environments. Java programs run on the Java Virtual Machine and are not dependent on a specific computer architecture.

4.1 Advantages to using Java

One of Java's strongest advantages is its object-oriented nature which encourages a modular style of coding and the ability to reuse code. Because it is such a well established language, Java has an incredibly extensive library adding to the power of the language. With regard to implementing DockAlt, Java's platform independence is a huge advantage as it allows Java programs to run the same regardless of the host machines. Java is designed to work well with distributed computing making writing network programs relatively easy. Additionally, it has an emphasis on security and reliability allowing for compile time error detection.

4.2 Disadvantages to using Java

Although Java is incredibly powerful and widely used, like the other programming languages, it is not with out drawbacks. For one, running Java programs requires having the heavy weight Java Virtual Machine creating overhead that is not an issue in other programming languages. With regard to implementing DockAlt, one overwhelming disadvantage to using Java is its lack of a Linux container binding. A possible work around to this issue is to use the Java Native Interface which allows the programmer to write native methods to handle the situation where the desired functionality of an application cannot be achieved with just Java because it may not support the necessary platforms needed, as is the case with Linux containers. While this approach will work, it requires extra, potentially challenging work to do what other languages such as Python and Go do out of the box. This work around to include Linux container support also negatively impacts the portability of the implementation, which is one of Docker's more advantageous characteristics. Additionally, the lack of features, such as dynamic typing featured in Python, can make programming in Java more difficult.

5 Rust

The Rust programming language was created by Mozilla Research and the open community and is designed to focus on the three main goals of safety, speed, and concurrency. Syntactically, it is very similar to C and C++; however, semantically it is quite different. Due to the goals it emphasizes, Rust excels in designing networking applications in addition to lower level programs.

5.1 Advantages to using Rust

Rust is a unique language that provides several advantages not featured in other languages. For one, as one of Rust's main focuses is safety, Rust includes numerous compile time safety checks that catch errors at compile time that other languages might not detect until runtime opening up the potential of a serious security vulnerability. By catching errors at compile time, there is little to no cost at runtime to having these security features. It's memory management system is quite efficient and ensures there are no dangling or null pointers without the need for an automatic garbage collector. Due to its guaranteed memory safety, Rust can run concurrent threads without having to worry about data races. Additionally, because Rust is relatively low-level with the majority of concurrency features being aspects of its library not the language itself, a programmer can implement an alternate method of achieving concurrency if Rust's implementation is not ideal for a specific application. Rust, like Python and Go, has Linux container bindings that when combined with Rusts safety, memory management, and concurrency characteristics, make Rust a strong candidate for implementing DockAlt.

5.2 Disadvantages to using Rust

Because Rust is a newer programming language, there are several disadvantages that come with its lack of maturity. Because of its extensive compile time checking and memory management, Rust tends to consume more memory and have large binary sizes when compared to other languages. Additionally, in some cases Rust can be lacking in performance. Being that it is a newer language with contributions from the open community, there are also bugs and a degree of unreliability in its libraries and packages. Rust also does not have the same support system that Python and Java benefit from.

6 Conclusion

All of the programming languages discussed in this paper have their respective advantages and disadvantages both generally and with respect to their viability as the language with which to implement DockAlt. Go, used to implement Docker, benefits from features such as static compilation, an extensive standard library, and support for Linux containers; however, it is relatively new, possesses bugs, and has limited support. Java is well established with platform independence and an extremely well-developed standard library. Although, despite its many strengths, with regard to DockAlt, Java has a large overhead as it requires the Java Virtual Machine and the Java Native Interface in order to make use of Linux containers. Python is a very flexible language with straightforward support for Linux containers, yet its lack of compile time checking and poor memory management system present potential issues. Finally, Rust benefits from Linux container support, is incredible safe, and handles concurrency and memory management well; however, it can suffer in performance and the immaturity of its libraries and packages. After examining the strengths and weaknesses of each language, I conclude that best choices for DockAlt are Python and Rust. As of right now, Python is likely the better choice between the two as it is a more well established language; however, as the language continues to develop it will become an increasingly well-suited language for this platform. Java is a poor choice due to the reliance on the Java Virtual Machine and the Java Native Interface.

7 References

[1] https://golang.org/doc/

[2] https://godoc.org/github.com/docker/libcontainer

[3] http://www.slideshare.net/jpetazzo/auth-doesnt-have-to-be-a-nightmare?next_slideshow=1

[4] http://sgros-students.blogspot.com/2013/04/lxc-python-bindings.html

[5] http://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html

[6] http://effbot.org/pyfaq/how-does- python-manage-memory.htm

[7] https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html

[8] https://www.docker.com/what-docker

[9] https://www.stgraber.org/2012/09/28/introducing-the-python-lxc-api/

[10]
http://www.streetdirectory.com/travel_guide/114362/programming/most_significant_advantages_of_java_language.html

[11]
https://www.ibm.com/support/knowledgecenter/#!/ssw_aix_71/com.ibm.aix.performance/advantages_java.htm

[12] https://doc.rust-lang.org/book/README.html

[13] https://www.codementor.io/rust/tutorial/steve-klabnik-rust-vs-c-go-ocaml-erlang

[14] https://doc.rust-lang.org/book/concurrency.html