An Analysis of Proposed Languages for DockAlt

Kacey Ryan
604047388
University of California, Los Angeles
Computer Science 131

Abstract

This report will analyze Docker and its development language, Go, in hopes of finding a language that can better replace Go in the implementation of a rewrite of Docker called DockAlt. The analysis of three alternative languages will be done. Python, Java, and Haxe are all possible alternatives in the implementation of DockAlt, each with its own special features it brings to the table. Java brings its speed and stability, Python lends its flexibility and simplistic style, and Haxe lends some specialized tools with a similar style to Go. In the end, it will be noted that Python lends the best hand in replacing Go for DockAlt. Although all three languages are viable options, Python's flexibility and extensive adoptability through its coding syntax and libraries leaves it unmatched.

1 Introduction

The proposed project is the management of a virtual operating system environment composed of Linux containers (LXC). The managing of this environment will be done with Docker. Docker utilizes the Linux container to isolate applications in their own chrooted virtual environment. In turn, Docker provides a runtime environment for these containers. It ultimately allows a smooth working environment complete with specialized versions of commands such as push, pull, commit, and run.

2 Docker and Go

Docker is written in the language Go. The reasons behind this decision break down into five major points as described by its developers [1]. The first revolves around the static compilation feature, which makes it an easier language right out of the door. It relies on no dependencies and can be a great bootstrapping tool. The second reason revolves around its neutrality. The fact that this language is not a popular one, its not Java or C++ or Python, makes it attractive. It takes the attractive features of all of these languages and makes them work effectively, leaving the bad behind and taking only the useful. This being said, it brings us to the third point, Go has the features Docker deems necessary. Go has asynchronous primitives, low-level interfaces, extensive standard library and data types, and powerful duck typing. The fourth point is an

interesting one, that is Go provides a full development environment providing commands such as doc (for documentation), get (for page fetching), test (controllable testing), and run (script prototyping). The final advantageous feature of Go is its multi-arch build [2].

The drawbacks of Go involve its average solutions to much more advanced problems. It solves numerous issues that pain the users but it doesn't necessarily solve them all in the best manner and thus some (John Mostovoy) would say, "it is not solving any problems". A design flaw of Go is its thread safety issues. The designers aimed for speed and thus developed a fast language but this requires a lot of attention on the programmers end to check thread safety. Go's test command has some annoyances such as not being able to run individual tests and not being able to run cleanup scripts. Some other issues include its lack of an IDE, its verbose error handling (considering all the programmer checks), and its odd lack of command line conveniences such as option handling.

3 The Task of DockAlt

Knowing these facts about the design, use, and implementation of Docker, we have been assigned the task of proposing the development of DockAlt. In the motion of designing a possible alterative to Docker, it has been proposed to find the most feasible language to write this DockAlt. The options available are Python, Java, and Haxe, each of which brings their own advantages and unique coding styles and system documentation to the table. A further analysis and comparison of these languages and their relation to Go will be done in hopes of finding the most appropriate replacement.

4 How About Java?

Java will be the first candidate up for assessment. The first thing analyzed will be the advantages to Java in relation to the development of DockAlt. Similar to Go, Java provides low level interfaces. This is advantageous in process control and more stable programming. Also adding to Java's weaponry is a great line of support and documentation behind it [3]. Java is well adopted in the programming community thus has a strong backbone and a lot to work with. The extensive libraries are similar to the strong standard libraries of Go, and thus make it a strong option. Of the analyzed languages in this report, Java has the best performance in most circumstances. Haxe and Python have small flaws that hold them back n the performance departments, and considering one of Go's primary objectives was speed, this is a plus. Another advantage of Java lies in its generated byte code, which allows for better optimization. Java does support multithreading in a well developed fashion and has asynchronous primitives much like Go. All of these advantages put Java up to a good start but now lets analyze the disadvantages.

Java's major difference and disadvantage when compared to Go is its lack of duck typing. Java has some of the strongest static typing available. This is not always a bad thing but in the case of DockAlt it is looked at as a disadvantage due to its lack of portability and versatility. It makes

code longer and more difficult to read and understand especially considering its already low level focus. When used advantageously, Java's static typing is safer, more stable, and a strong tool for reliable programs. Another disadvantage of Java is its lack of neutrality. It s a C derived language and acts like one in every sense. The syntax is difficult to pickup and built for stability not the versatility that is preferred for Docker. Go was decided for the reason that it contained a few uniquely specific features, which most languages including Java lack such as a full development environment.

5 How About Python?

The next language to be analyzed is Python. Python is expected to be a fairly competitive candidate considering its similarity to Go in many features. Its goal as a language is versatility and flexibility as well as simplicity [4]. Where Java was a strong candidate that lacked the flexibility, Python will be the strong candidate that lacks the speed aimed for by Go. The advantages of Python are numerous considering it shares the same strong duck typing as Go, has simple syntax, extensive libraries to draw on, great community support and documentation, and it is a great cross platform candidate. Python provides multithreading that is not as well designed as that of Java, however it also provides for asynchronous primitives through asyncio. The python language is interpreted which comes in handy for easy development but due to the duck typing, Python is prone to runtime errors however this is something that I'm sure Docker is already used to because of Go.

The negative aspects of Python lie in its lack of neutrality and speed. Python is similar to Java in the case that it is a well-adopted language that does not cover every aspect that Go covered. It misses the unique advantages mentioned previously with Java, but it also lacks the low level primitives of Java and Go. This lack of control limits the potential speed of Python and thus hinder Go's goal of fast computation at the programmer's expense [5]. Python as mentioned previously has limited support for multithreading and thus is not the best candidate for this sort or usage. The detail, however, that keeps Python as a strong candidate is the opportunity for speed improvements. The fact that Python is not as fast as Java is only viable in specific areas. Python has the speed of development as well as competitive performance speeds when compared to most languages [2].

6 How About Haxe?

The final language that will be analyzed is Haxe. Apart from anything, Haxe has neutrality on its side. It is not Java, it is not Python, but it's a mixture of features, similar to Go. With that being said let us observe the advantages of utilizing Haxe to develop DockAlt. Haxe supports strong duck typing similar to Go and Python [6]. This advantage provides a large number of types to choose from. The code for Haxe is in between the simplicity of Java and Python as it is moderate in the field of syntax understandability. The unique features of Haxe lie in its cross platform abilities

and its cross compiler. This makes some parts of Haxe more complicated and slower, however it is advantageous if used correctly. Haxe also provides a multithreading interface with asynchronous APIs. Haxe is very similar to Go, but it does not provide all of the perfection that we look for in hopes of implementing DockAlt.

The disadvantages of Haxe are numerous. First off, Haxe lacks community support and extensive documentation. Its libraries are nowhere near as extensive as Go, Python and Java, which leave it at a major disadvantage in robustness. The main feature it provides is the cross compiler which is a useful tool, but may not be as advantageous as originally thought as it houses many exceptions which have to be accounted for in coding. This hurts performance as well as simplicity of development. Finally, Haxe is not as reliable of a language. Java provided the best stability of all of the languages mentioned, Python provides the most flexibility, and Haxe is merely in between. In fact, average is a good word to describe Haxe. It is similar to Go in the fact that it solves no problems, however, it lacks the solutions that Go provides for the important problems.

7 The Conclusion

The most feasible option in my opinion in order to develop DockAlt is to adopt Python. Overall, Python is the most similar to Go in its flexibility, robustness, portability, and simplicity, which is the primary goal of Go. It lacks the speed that is preferred, but this is something that can be worked around. Java would be a close second and Haxe will take the last place spot. The reasoning for this is the fact that Python and Java provide competitive solutions for the problems they are meant to support. For instance, Java is one of the fastest languages of the ones listed and it dominates the field in code reliability. Python is the most favorable to developers and its flexibility is unmatched. Haxe has some great features but none that are necessary and unmatched by other languages. Overall, all of the languages provided could be utilized in the creation of DockAlt, however, Python's advantages are the most fitting for Docker's primary goal of flexibility.

8 References

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

[2] http://golang.org/doc/

[3] http://www.oracle.com/us/technologies/java/resources/index.html

[4] https://docs.python.org/3/index.html

[5] http://twistedmatrix.com/users/glyph/rant/python-vs-java.html

[6] http://old.haxe.org/doc/why