Analysis of Docker Implementations in Various Languages

Omar Ozgur, *UCLA*

Abstract

Docker is a platform that allows for software to be managed and distributed through the use of lightweight standardized Linux containers. This paper examines the feasibility of creating DockAlt, an alternative to Docker, in the Java, OCaml, and Dart languages.

1. Introduction

The Docker platform is widely used to ease the process of managing and deploying software. Instead of creating separate virtual machines for applications, Docker uses Linux containers to package application dependencies, including libraries, namespaces, and cgroups. This allows for multiple containers to share a single kernel, which reduces the traditional overhead of running applications that were developed in various environments.

Although Docker has become a useful tool for many developers, there are some issues with the platform that should be considered. One problem is that bugs in the main version of the software may cause numerous deployment issues due to the fact that there is a single source for Docker. Other issues may arise due to the use of the Go language, which are explained in further detail in section 2. In order to alleviate some of these problems, other languages may be considered in order to implement DockAlt, an alternative to Docker. Each of the languages discussed in this paper have features that would improve or diminish specific features of the Docker platform.

2. Go

Go is a relatively new language that was announced by Google in 2009. It aims to be "expressive, concise, clean, and efficient", as stated in the official documentation. It focuses on providing extensive support for concurrency, as well as for modular code. It uses features such as static typing and compilation to make code fast, while it also attempts to make the development process simple and efficient.

2.1 Advantages of Go

Static compilation in Go makes the language a good candidate for the Docker platform. This allows for software created in Go to remove the need for most external dependencies. Although static compilation usually increases the sizes of compiled applications, it makes code more portable since dependencies are included within executables. This is a benefit for Docker, which is built to make deployment easy across various machines.

The Go language also has built-in support for many powerful features. Asynchronous primitives allow for developers to easily avoid blocking operations. Low-level interfaces give programmers the ability to manage system resources efficiently. Extensive libraries and data types make it easy to implement a variety of complex features. Strong duck typing improves development flexibility when using operations on various data types. The language also makes it easy to build applications for various platforms, which is a great benefit for Docker. Go also has a powerful development environment. Commands are provided to do common tasks such as look up documentation, fetch project dependencies, and change code formats. This improves development time and removes the source of much unnecessary repetition.

2.2 Disadvantages of Go

Although many of the features of Go are great for the purposes of Docker, there are some disadvantages to using the language. One obvious disadvantage is that the language is relatively new, and not as well-known as other languages like C and Java. Many other languages have similar features as Go, so it may make sense for a system like Docker to be built in a language that most developers are more accustomed to.

Another disadvantage, which was highlighted in the slides posted by Jerome Petazzoni, is that maps are not thread-safe in Go. This allows for threads to have great speed when accessing data in maps, but it also allows for errors to be introduced without manual synchronization. This increases the complexity of code, and makes it more difficult to create fast programs while also ensuring correct results, which is important for a system like Docker.

Another feature that is not available in Go is a full IDE. Although simple text editors and extensions can be used to control a programmer's development environment, a powerful IDE would give better access to powerful features such as debugging tools and documentation.

3. Java

Java is a popular object-oriented language that can be used to create powerful platform-independent code. It's popularity and flexibility make it an interesting candidate for implementing the DockAlt platform.

3.1 Advantages of Java

From a development standpoint, Java's widespread use has led many developers to learn the language. Extensive libraries can be used to implement complex features, and Java's object-oriented nature can be used to create clean and modular code. Although Java's static typing can lengthen the development process, it can also improve the detection of errors, which can be beneficial for a system that relies on robust code. The fact that code is compiled to bytecode for the Java Virtual Machine (JVM) allows for code to be platform-independent, which is useful for creating a system like DockAlt. This also improves the speed of applications, as opposed to using an interpreted language.

Java's popularity, platform-independence, compiled nature, and reliability are advantages that should be considered for the DockAlt platform.

3.2 Disadvantages of Java

Java's heavy reliance on the Java Virtual Machine forces systems to include the JVM if they wish to run developed applications. This can lead to heavier-weight systems, which mitigates some of the effectiveness of the traditionally lightweight Docker platform.

Another disadvantage is the lack of support for Linux container bindings in Java. One workaround is to manipulate containers through the use of provided command-line interfaces. This may work, but it is often difficult to ensure that these interfaces are correctly used on individual platforms. Another option is to utilize the Java Native Interface to call libraries from other languages in order to control Linux containers. However, this has the similar problem of breaking native platform-independence.

Additionally, the syntax and static typing of Java often leads to the creation of more complicated code. Although this reduces errors, it can also increase development process. This could potentially increase the amount of time that it takes to develop the DockAlt platform as opposed to using other languages.

4. OCaml

OCaml is a powerful functional programming language that focuses on "expressiveness and safety", according to the official website. Its ability to produce fast machine-independent code makes it another candidate for the DockAlt platform.

4.1 Advantages of OCaml

Although OCaml can be compiled to native platform-specific code using a compiler like "ocamlopt", it can also be compiled to bytecode with a compiler like "ocamlc". Compiling to machine-independent bytecode is an advantage for a system like DockAlt since applications can be ported to various systems easily.

OCaml also includes many powerful libraries and high-level constructs that make it easy to create complex features. Libraries are also provided to perform low-level tasks like managing system resources. Similar to other languages, features like automatic garbage collection and static type checking reduce the number of errors that are introduced into applications. OCaml's high-level abstractions, powerful libraries, and portable compiled bytecode would help to make the DockAlt system reliable and robust.

4.2 Disadvantages of OCaml

Since OCaml does not currently have a native binding for Linux containers, a workaround would be necessary for the DockAlt system. One method would be used to use an 3rd party library to manage Linux containers. Another option would be to use the LXC command-line interface to deal with containers directly, although this may lead to issues on separate platforms and Linux container versions.

Another disadvantage is that many developers may not be familiar with OCaml development, or functional programming languages in general. Since the language is very different from other popular languages like Java and C++, it may take more time for developers to learn it well enough to create a robust platform like Docker.

One other disadvantage is the fact that OCaml bytecode is usually slower than compiled platform-specific code. This is mainly because the compiler cannot make specific assumptions that would improve execution speed on any specific platform. The fact that the bytecode needs to be run in a virtual machine can also add unwanted overhead to systems.

5. Dart

Dart is a relatively new programming language that was announced by Google in 2011. According to the official website, Dart focuses on being a language that's "easy to learn, easy to scale, and deployable everywhere." These important features make Dart another possible candidate for the DockAlt platform.

5.1 Advantages of Dart

In terms of application development, Dart has constructs that aim to make development fast and easy. Built-in tools and libraries provide powerful abstractions that deal with common programming problems, and give programmers the ability to deal with low-level system resources. The developers of the language also state their intent to keep it reliable and stable. These goals are seen in features such as static type checking and error detection, which help to ensure that applications don't fail. Another advantage of Dart is that is can be compiled to JavaScript using a compiler like dart2js. Since Dart is a relatively new language, this feature can increase adoption and support of Dart applications. These features would all help to make a reliable DockAlt system.

5.2 Disadvantages of Dart

The DartVM virtual machine is typically used to allow for Dart application source code to be run on various machines. Although this does improve system compatibility, it forces systems to include the virtual machine if they wish to run Dart applications, which can introduce unwanted complexity. An alternative to this is to compile Dart code to JavaScript code, which can be run in a common JavaScript engine.

Additionally, there are no official Linux container bindings for Dart, as seen with the Java and OCaml languages. In order to add support, 3rd party libraries would likely need to be used or developed. Furthermore, Dart is a relatively new language compared to Java and OCaml. For this reason, many developers may not be familiar with it, which could lead to longer development times for the DockAlt platform. Recent additions to the language could also potentially cause bugs to be added to the platform.

7. Conclusion

Java, OCaml, and Dart all have advantages and disadvantages in terms of creating the new DockAlt platform.

Java's popularity and powerful libraries allows for robust code to be developed with relative easy. However, the required virtual machine adds overhead to systems that wish to run Java applications. Additionally, the workarounds needed to obtain Linux container support could cause application to become buggy, or even lose platform-independence. These disadvantages would likely make it more difficult to build the DockAlt platform in Java than in OCaml or Dart.

Although OCaml is not as popular as Java, it does have extensive libraries that provide abstractions for difficult tasks. Its focus on code safety helps to catch bugs before applications are compiled. However, similar to Java, OCaml bytecode needs to be run in a virtual machine, which can add overhead to systems. The lack of Linux container support also leads to the need for workarounds to be used. It may also be difficult for developers to develop large applications with OCaml if they are not familiar with functional programming. Provided that support for Linux containers can is found or developed, OCaml would be a decent choice for the DockAlt platform compared to Java and Dart.

Dart is a relatively new language, but it does have many constructs and built-in libraries that help when dealing with difficult tasks such as synchronization and error handling. Like Java and OCaml, Dart needs a virtual machine to run on a variety of platforms, which can add system overhead. Dart also does not have official Linux container bindings. However, the ability for Dart code to be compiled to JavaScript could possibly allow for it to use LXC bindings that are available for certain JavaScript frameworks.

Although none of the three languages that were examined are perfect for the purposes of creating an alternative to Docker, Dart seems to be the best choice. Although it is a relatively new language, its powerful libraries and its close association with JavaScript can give it the low-level support that is needed to create the DockAlt platform.

8. References

[1] Dart. "A Tour of the Dart Language." https://www.dartlang.org/guides/language/language-tour

[2] Docker. "What is Docker?" https://www.docker.com/what-docker

[3] OCaml. "Compiling OCaml projects." https://ocaml.org/learn/tutorials/compiling_ocaml_projects.html

[4] Paolucci, Nicola. "Static Go binaries with Docker on OSX." https://developer.atlassian.com/blog/2015/07/osx-static-golang-binaries-with-docker/

[5] Petazzoni, Jérôme. "Docker: An inside view." http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/23-Why_Go4_full_development_environmentGo

[6] Seguin, Karl. "Go's error handling: good and bad." http://openmymind.net/Golangs-Error-Handling-Good-And-Bad/

[7] The Go Programming Language. "The Go Programming Language Specification." https://golang.org/ref/spec

[8] Wang, Chenxi. "Containers 101: Linux containers and Docker explained." http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html