COM SCI 131 Homework 6
Containerization Support Languages Report

Weifeng Jin, 904429456
University of California, Los Angeles

November 30, 2016

Abstract

Docker is a open-source software containerization platform developed by Solomon Hykes that automates the deployment of Linux applications inside software containers. As described in its own website, the main goal of Docker is to "Build, Ship and Run Any App, Anywhere". Originally Docker is implemented in Go, which is a relatively new free and open-source programming language created at Google in 2007. This report examines the possibilities of the implementation of Docker in OCaml, Java and Dart to achieve the same functionality and performance goals. Also, in the report I would try to analyze the advantages and disadvantages of each version of the implementation for the application.

Keywords: Docker, OCaml, Java, Dart, Programming Languages

1. Introduction

The main motivation of the development of Docker is to introduce a solution to the deployment problem in the Linux Container (LXC). Generally speaking, Docker creates a standard format for containers and a place to share them. Docker uses a high-level API to offer lightweight containers and run processes in isolation. The typical use of Docker is as follows: 1) fetch some data from the registers, 2) access the data and make some changes of it, 3) record these changes in our own database, repeat as many times as we want, 4) share the resulting file on a public register or a private register.

2. Docker in Go

Before we consider other possibilities of the implementation of Docker, we should really examine the choice of the development team of Docker. The programmers in the development of Docker chose to use Go as their programming language, which is a quite new language at that time (even today) and they have listed several reasons in their inside talk about why they chose Go to develop the software platform.

2.1 Neutrality of Go

One of the reasons that programmers chose Go is that they need a neutral language to build the application. It cannot be C++, Python, Ruby or Java since for any one of these popular languages, there is a large community of opponents and people are not easily getting accustomed to them. Also, Go has many great features that can support the implementation of Docker, including good asynchronous primitives, static compilation and low-level interfaces. There are some other features of Go that is suitable for the design and implementation of Docker.

2.2 Static Type Checking and Scalability

Since Go is a programming language with static type checking, it is scalable to large systems which is greatly appreciated nowadays. Nowadays, commercial softwares have a lot of choices in platforms, each with its own libraries and dependencies, a easy scalability ensures that the programmers can easily transfer from what they had written (their first version of the project was written in Python) to a new platform.

2.3 Easiness to Read and Write

Also, Go is pretty readable and easily-maintainable and fast compilation due to the lightweight type analysis and easy dependency analysis. Since the programmer need a good memory/package management system, and easiness to write/read, they chose Go as their development language.

3. DockAlt in Java

Java is one of the most popular and widely-used programming languages in the world. Personally speaking, Java is mainly an imperative and object-oriented programming language. Java does have some great features that are perfectly suitable for the development of DockAlt, but overall it is not a suitable programming language for the design of the Docker application.

3.1 Static Type Checking

Since Java is a strongly and statically typed language (like Go), Java's typing system improves safety and reliability of DockAlt. A strong type checking helps the compiler easily detect mistakes and report error during compile time. Also, Java has a garbage collector and proper memory management system to improve the platform portability for Go. Java programs can run on different platforms because most platforms have a dedicated JVM. This feature is also similar to Go, and is important to the development of Docker since one of the initiatives is that Docker can work on various platforms to improve scalability.

3.2 Verbosity of Java

However, Java is way complicated than Go and the need of Docker's programmers. The verbosity of Java might increase the difficulty of adoption of new users of Docker and is not a good choice for rapid prototyping purpose. The goal of Docker is to attract a large number of users and build a huge community, so the programmers need an easy-to-write/read language rather than language as complicated as Java. Also, the complicated, multi-leveled hierarchy types in Java might slow down the speed of compilation.

3.2 Linux Container API Support

Another drawback of Java is that it lacks local support for Linux Containers API. The current Linux Container API support in Java is provided by some third-party libraries. It would be not so convenient for the developers to figure out the correct use of the new API from some unreliable online resources and adds up to the development difficulty.

4. DockAlt in OCaml

OCaml and Go are two quite different languages, but they have some similarities. OCaml and Go are both doing strong and static type checking, which have been discussed above as a valuable advantage of the programming language. The strongly and statically typing in OCaml ensures a safe and reliable implementation of Docker. In this sense, OCaml offers a safer implementation over Go since Go lacks compile-time generics which might result to several serious problems during compile time.

4.1 Functional Programming

In terms of language syntax, OCaml is also a concise language, which is similar to Go has. The programmers in the team of Docker need simple languages to work with rather than complicated languages like Java and C++.

However, due to the nature of a functional programming language, OCaml is not a language that is friendly to beginners. OCaml is not easy to read for those unfamiliar with function programming. Nowadays, most languages for education use a style of object-orientation. Although Go is not a purely object-orientation language, it has a lot of similar features as these languages and are more easily-understand by beginners than OCaml.

4.2 Linux Container API Support

Lacking support for a Linux Containers API is another drawback for OCaml. Similar as Java, it is not convenient for the developers to figure out how to use

Linux Containers API without built-in library support in a programming language. Also, in terms of performance, generally Go delivers a better performance than OCaml in this kind of application since it has a better memory allocation mechanisms and less recursive calls than OCaml.

Overall, OCaml is not a suitable candidate (or a better candidate than Go) in terms of the development of Docker.

5. DockAlt in Dart

Similar as Go, Dart is a general-purpose programming language developed at Google. It is a class-based, single inheritance, and object-oriented language. The main application of Dart includes web, server and mobile applications.

5.1 Neutrality

Dart relies on a source-to-source compiler to JavaScript. The Dart code is precompiled into JavaScript using the dart2js compiler. This violates the neutrality principle that programmers in the team of Docker wants to obey. Since Docker is aimed at a portable application in a large scale of platforms, such a limit in portability is not ideal for the development.

5.2 Optional Typing

Dart implements using an optional typing system, which allows a quick dynamic typing like Python but it allows the system to add type checking and reduce runtime errors. Similar as Go, Dart follows a concurrency paradigm. It uses isolates to achieve concurrency. Since threads are processed in isolated memory heaps, if one container crash, it would not influence other containers in Docker since they are isolated. This feature improves the safety and reliability of the application.

5.3 Linux Container API Support

Dart also lacks Linux Container API support, which is a significant advantage in terms of the development of Docker, which relies heavily on Linux Container applications.

6. Conclusion

After considering the original language design choice (Go) and other three possibilities (Java, OCaml, Dart), I would say Go is still the best candidate for the development of Docker. Having a builtin Linux Container API support is a great plus and huge benefit for the programmers. Java is way too complicated and OCaml lacks a lot of great important features for the application. For the three alternatives, I would argue that Dart is a great one. However, Dart is a very new language (it is even younger than Go). It lacks a mature development community and detailed documentation in many important aspects. Currently, I would say Go is till our best option for applications like Docker.

7. References

[1] Docker website: https://www.docker.com

[2] Fall 2016 COM SCI 131 Homework 6 Specification: http://web.cs.ucla.edu/classes/fall16/cs131/hw/hw6.html

[3] Docker: an insider view: http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go

[4] Docker: Automated and Consistent Software Deployments: https://www.infoq.com/news/2013/03/Docker

[5] Wikipedia page description of Dart: https://en.wikipedia.org/wiki/Dart_(programming_language