Containerization Support Languages

Tianyang Zhang 404-743-024

Abstract

LXC (Linux Containers) is an operating-system-level virtualization method for running multiple isolated containers on a control host using a single Linux kernel. Docker provides additional layer of abstraction and automation based on LXC. It helps to avoid the overhead of starting and maintaining virtual machines. Docker is initially released on March 2013. Since it is so young, if the main version is buggy, the whole program based on Docker would be buggy, even crash. Therefore, the purpose of this report is to find a suitable language to re-write an alternative Docker called DockAlt.

1 Introduction

Containerization is also called container-based virtualization, is an OS-level virtualization method for deploying and running isolated applications without running virtual machine entirely for each application.

Containers are run on a single control host and accessing a single Linux kernel in LXC. Linux Containers spends less resource comparing to virtual machine since it is implemented base on Linux kernel, and the cost of implementing by kernel is relevantly much less. Linux Containers could also use to isolate specific application. It could provide the virtual environment to the application in a sand box with little cost and fast speed, where traditional virtualization would need create virtual machine, launch the system, then launch the application.

Docker is an open source engine which could easily create a lightweight, movable, and isolated container. Docker could be used in web application automate packaging, automation testing, or extend exist OpenShift or Could Foundry platform to create your own PaaS environment.

Here we are going to discuss the easiness, compatibility and several other characteristics of several programming languages to re-write the Docker.

2 Go

2.1 Advantage of Go and why using Go

Docker is written in Go, which is also a new language that released on November 2009. Go is a concurrent and fast-complied programming language with garbage collection. A large Go program could be compiled in seconds. For Docker, there are several reasons why the choose Go to write it.

2.1.1 Static Compilation

The ‘go build’ command of Go embed almost everything they need for the application, rather than install lots of third party packages to achieve the goal. Thus, the application would be easier to install, easier to test, and easier to be use. Static compilation is also ‘safer’ comparing to dynamic since it compiles the Go code to machine code before execution.

2.1.2 Features Provided / Neutral

Go provides features exactly needed by Docker. Such as asynchronous primitives -- which used a lot in proxy herb, low-level interface, and extensive standard libraries and data types. It also has strong duck-typing as Python, which this would makes the develop of Docker easier. Furthermore, Docker is more likely to be adopted due to Go’s neutral, since there is no bias against it.

2.1.3 Muti-Arch build & Development Environment

Go is platform independent, it could build without the need to pre-process. And Go addresses multiple issues of the development workflow, such as ‘go doc’, ‘go get’ etc.

2.2 Disadvantage of Go

First, the garbage college system of Go is too simple. Go just use the original mark-and-sweep garbage collector. This garbage collect system would stop the whole program when doing the GC progress, which is much less efficient comparing to lot of other programming language. The other one is generic. There is no generic in Go, which means there will be a lot of very similar but repeated method inside the application, which makes the code harder to read and wastes developing time. Programmers may use 'interface {}' to reach the similar feature as generic, however, it is similar to 'void*' pointer in C, which cannot be checked by static type checking, and thus may cause problems that hard to find.

3 Java

Java is an old programming language comparing to Go, it is tested by huge number of programmers and applications. As one of the most famous Object-Oriented Programming language, Java has a lot of shining points.

3.1 Advantages of Using Java for DockAlt

Java uses static type checking as same as Go. The debug process would be easier for programmers by using static type checking, and also safer to run comparing to dynamic type checking. Another advantage is Java has generic types, in this case, it could save programmers a lot of time by merge multiple similar and repeated Go methods into one Java class with less code, which could improve the re-writing efficient.

Moreover, Java is a relevantly network-strong language, it provides high secure with plenty of safety checking such as array index checking, Bytecode checking. Comparing Java to Go, garbage collecting system, and the exception handle system of Java are also very strong.

Finally, we could also use Java to build a high concurrency application which remedy the low asynchronous of Java. Using Java to re-write Docker could still get good performance.

3.2 Disadvantages of Java

One of the biggest disadvantage of Java is its design patterns. There are 23 commonly used Java design patterns, and a lot more not used very often, this makes the coding in Java harder than Go. Comparing the program of Java and Go, Java's program structure would be more complex.

Java also has a massive library for external modules, although this provides programmers more tools for use, it makes programmers needs to learn much more than Go.

4 OCaml

OCaml is a functional programming language which combined abilities of both object-oriented language and functional programming language.

4.1 Advantages of Using OCaml for DockAlt

Ocaml is a strong statically typed language. The type inference is done by compile-time binding. This allows programmers to type much less amount of type names as long as the operations done on the data is ensured correctly.

Also, unlike Go and Java, OCaml has abilities for both Object-Oriented Programming and Functional Programming. As a functional programming language, OCaml does better on asynchronous methods than Java.Thus the DockAlt in OCaml may behave better performance since we are going to use it as the framework for proxy herds.

Furthermore, OCaml is compiled down to the bytecode as same as Java, which means it would be highly portable with different machines or operating systems.

4.2 Disadvantages of OCaml

OCaml does not allow polymorphism (only let-polymorphism), which is same as Go. This could be one disadvantage comparing to Java, especially for programmers who get used to Object-Oriented programming language such as C++/Java/Python. Programmers who never used functional programming language may could also have a hard time at the beginning of this project.

The other serious disadvantage of OCaml comparing to Java is that OCaml doesn't have a high complete and strong library to binding with Linux Containers. This means programmers needs to start from almost zero to build up this entire application.

5 Rust

Rust is a system programming language which developed for multi-core system. It emphasizes safety, performance, and concurrency.

5.1 Advantage of Using Rust for DockAlt

One of the advantage for Rust is that it has deterministic destruction, which means it has much better performance on garbage collect than Go and the garbage collection happens parallel with main process.

Rust is a strongly static typed programming language as OCaml, it has the same advantage such that the code relevantly safe. However, Rust is even more safer, there is no null pointer, and programmers doesn't need to care about memory management. Rust also pulled in concepts like borrow checker to ensure more on safety. (*The safety here means it has very little probability crash in runtime)

The data type system of Rust is also more extensive. Programmers could use Rust to implement high level abstraction and more extended libraries, this is a big advantage on future development by using Rust.

For the last point, the developer community of Rust would love to make changes, but the developer community of Go always insist their own thought (such as they consist Go doesn't need generic). This makes Rust more flexible and more potential in the future.

5.2 Disadvantage of Rust

The most obvious disadvantage of Rust is that it is relevantly harder to learn. The syntax of Rust looks like the combination of C++/Java with OCaml with self-extension. Rust allows you to extend syntax that not built-in with Rust by using '!' keyword, and there are a lot more concept such as borrow checker or lifetime etc. These makes programmers harder to get start with Rust.

The other disadvantage is that Rust is less mature than other 3 languages, and it's less commercialized. The third-party packages for Rust are also much less, and thus Rust is less stable than other 3 languages mentioned above.

6 Conclusion

All of those alternative language discussed above have the ability to re-write Docker to an application with good performance, and they all provide strong concurrent paradigm.

For language easiness, Java should be the easiest to learn and handle among 3 of those languages. For safety and future development, Rust seems to be the one that has lowest probability on program failure, and it would be the one developing and growing the most in the future. For coding efficiency OCaml seems to be the one with lighter code and fast to write if programmers are willing to get through the initial learning.

There is no such best choice among those three languages for re-writing Docker, I would suggest the company to analyze what advantages we need the most, and what disadvantages we need to get rid of the most. Then we can choose the one that fitting the company's purpose the most.

7 References

1. Docker documentation
https://docs.docker.com/engine/reference/commandline/system_events/#limiting-filtering-and-formatting-the-output

2. Linux Containers
https://linuxcontainers.org/lxc/

3. Docker and Go: why did we decide to write Docker in Go?
https://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/22-Why_Go3_it_has_what

4. Java Garbage Collection Basics
http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

5. Go (programming language) – Wikipedia
https://en.wikipedia.org/wiki/Go_(programming_language)

6. Java (programming language) – Wikipedia
https://en.wikipedia.org/wiki/Java_(programming_language)

7. OCaml – Wikipedia
https://zh.wikipedia.org/zh-hans/OCaml

8. Rust Documentation
https://www.rust-lang.org/en-US/documentation.html

9. OCaml for the Skeptical - Strong Static Typing with Type Inference
http://www2.lib.uchicago.edu/keith/ocaml-class/static.html