DockAlt: An alternative of Docker

Ziheng Zhou
University of California, Los Angeles
Jun 5, 2015

Abstract

We would like to run our projects on a large set of virtual machines. In order to keep costs low, we decide to standardize on Linux kernels and use operating system-level virtualization with Linux containers. And we decide to use Docker platform to achieve this goal. Docker is written in Go, which is a concurrent language created in Google. However it's a very young language, so it's not mature yet and may cause big trouble if it has bugs. We don't want to put all the eggs in one basket, so here we are exploring the possibility of implementing Docker using another language. Let's call it DockAlt.

1. Docker

1.1. Overview

Docker is a lightweight Linux container virtualization platform with workflows and tooling for easy managing. Developers and sysadmins can build, ship, and run distributed applications with it. Docker can help you separate your applications from your infrastructure and treat your infrastructure like a managed application.

1.2. Core of Docker: Linux Container

At the core of Docker is Linux container. Linux container is like a virtual machine but doesn't have a guest operating system inside, which makes the package heavy and slow. Instead, it uses the Linux kernel of the host machine and just includes the binary dependencies of the application. Therefore it's super lightweight and gained much better performance and makes shipping much faster. Docker has its own container format, libcontainer, but it also supports traditional Linux containers using LXC.

1.3 More features of Docker

Docker utilizes a technology called *namespaces* to isolate containers from each other. With a different set of namespace, each container does not have access to any resources outside.

Docker also makes use of *cgroups* or control groups to share available hardware resources to containers, and if required, set up limits and constraints, like the memory available to a specific container.

Beside, Docker also uses Union file systems, or UnionFS, wchi are file systems that operate by creating layers, making them very lightweight and fast.

2. Go

Go is a statically typed and compiled language that feels like a dynamically typed (duck typing), interpreted language. This makes projects not only easy to install and test but also easy to write too. And also it requires no dependencies. "Go build" will imbed everything you need. Also it is a very light and fast language. For low level programming, like Docker that relates to linux kernel hacking, it is a good advantage. The low level interfaces of Go, like manage processes are system calls are very nice too. Go also addresses multiple issues of the development workflow with simple commands like "go doc" (for looking up documentation), "go get" (for fetch dependenceies on github etc), go test (runs all test functions) and so on.

Most importantly, it has built-in concurrency primitives like goroutines, which is an abstraction of a lightweight thread, or multiple threads if needed, and channels, which provides ability for different goroutines to communicate and synchronize with each other. And the select key word can have goroutines to avoid hanging situation by allow it to either receives data from channel or time out and keep going.

However, Go lacks compile-time generics, which leads to repetition or excessive dynamic code. And the maps are not thread safe.

3. Java

Java is generally a great language for most tasks, but a horrible fit for implementing Docker. It has great objective oriented design, but unfortunately Docker doesn't need this feature that much. It is dynamically compiled, but unfortunately this is exactly not what Docker wants. Dynamically compiled language makes it almost impossible for compilers to figure out dependencies, not to mention install dependencies for you. Java does not have duck typing feature, a loss for the ease of programming. Java uses Java Virtual Machine (JVM),

which enables Java to run on any machines. However, this feature is a total redundant feature for Docker. Docker engine only rests on top of Linux Kernel. It will not talk to any other machines beside Linux. So having JVM only slows down the execution and does no benefits. Beside, Java doesn't support LXC binding, which is a huge loss too.

4. Python

Docker was orginally prototyped in Python and later rewritten in Go. This mere fact not only shows it's possible to write Docker in Python but also show Python is a good alternative (don't forget we want an alternative not for it to perform better than the current one, but simply for having an alternative that's reliable).

Python has strong duck typing, so it's easy to write. Also it's strongly typed, forbidding operations that are not well-defined rather than silently converting types to fit into the operations. This is desired in Docker because this helps write more robust code. Also Python has great readability, making it easier to write and collaborate on. More importantly, Python is a very mature language with strong community and a wide range of libraries. Therefore it's very unlikely to meet bugs of Python itself, and when you have problems you are much easier to figure it out.

But the fact that the developers switch out of Python shows there're some downsides of using Python. First, Python is a dynamically typed language, which makes the system less able to test. And also it's an interpreted language, so it's a lot slower than Go. For low level structure like Docker, this is an important feature. Aside from these, Python is also a high-level language, which abstracts a lot of details of low-level details, giving less flexibility for writing low level functions.

5 Groovy

Groovy is a multi-faceted language for the Java platform. It has many powerful features. First, it is optionally typed, meaning it can be both static-typed and dynamically typed. With this feature we can gain the static type feature that we desired for Docker. Second, it is also an optionally dynamic language, meaning it has static compilation capabilities. So it has static compilation feature that is a huge win for Docker, since it can help docker to figure out dependencies easily. Third, it has a concise, readable and expressive syntax, making it very easy to learn. Fourth, it has a vibrant and rich ecosystem, including concurrenty / asynchronous/ parallelism library, which is a very desired feature for Docker, just like Go. And it also has test frameworks, and build tools that can help development a lot. Finally, Groovy can also be used as script, which makes testing very easy and straightforward.

The only downside is that it's built on JVM. So the code will be compiled to Java bytecode first, making it much slower than necessary. Also because JVM doesn't support LXC binding, which is a huge loss for Docker.

6 Summary

Comparing all the languages above, the best alternative would obvious be between Python and Groovy. Groovy has a lot of great features that can be very helpful such as static compilation and scripting. But the goal we want for DockAlt is not the best language, but the realiability and ease to develop. Python has better community, is more mature, easier to write and collaborate on, and it supports LXC binding. In this sense, I would choose Python. Afterall, the prototype of Docker was written in Python.

References

[1] http://groovy-lang.org/

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

[3] https://linuxcontainers.org/lxc/

[4] http://jcsites.juniata.edu/faculty/rhodes/lt/plcriteria.htm

[5] https://www.docker.com/