Homework 6 Containerization support languages

Bolun Hu

University of California, Los Angeles

Abstraction

In this paper, I discuss the feasibility of implementing Docker using an alternative language among Java, Python, and Dart. Three languages will be compared in their suitability to replace the Go language. In the end, Python seems to have the best suitability in implementing Docker based on its easiness in coding, strong duck typing, and its extensive standard library and data types.

# 1. Docker

## 1.1 What is Docker?

Docker offers us an envelope, or a container, that can run our applications. Docker is written in Go programming language. It is something like adding pipelines onto the LXC (Linux Container), therefore allowing users to work on higher level concepts.

Docker extends the LXC. Through a high level API, it offers threads with a light weight virtual environment. Docker takes advantage of the LXC, cgroups and Linux kernel. But different from a traditional virtual machine, Docker does not contain a separate OS, but instead operate based on the given infrastructure and functionalities that current OS offers.

Docker works like a portable container engine. It packs up the application and all other dependent environment that the application relies on into a single virtual container. This virtual container is runnable on any types of Linux server, thus largely improving the application’s portability and flexibility.

Docker is consisted of several components: Docker Server Daemon, for managing all containers. Docker command line client, for controlling server daemon, and Docker images, for searching and browsing docker container images

## 1.2 Why Docker? We have VMs already aren’t we?

Yes. But VMs is a little bit low class and resource consuming compared with Docker. When transplanting programs through VMs, it takes a long time for VMs to initialize different kinds of hardware components such as drivers, virtual processors, network sockets, etc. But those components may even be useless for most of the time. At the same time, VMs will allocate a lot of system resources like memory and CPU.

On the other hand, Docker is light weighted and faster. It runs like a normal program, with faster speed and much less resources allocated. That is why Docker is a valuable option.

# 2. Go Programming Language

## 2.1 Why Go?

Static language but with dynamic feeling

With static compilation, most hidden problems will be detected during compile time. Go also gives users a lot of useful packages to choose from, which makes programmers more efficient and easier to write code. Moreover, from the interview slides, “go build” embeds everything we need. (Go does not have inheritance) Also it is easy to install, test and adopt.

What about Python and Java?

Python is a dynamic language, which means some optimizations will not be involved until run time. Thus the start-up time for dynamic languages like Python will be a disadvantage here.

Java as we know, is a static type language. Thus it should be considered as a qualified substitute in this aspect. However, when comparing the speed of Java compilation and Go compilation, it turns out that Go still wins by a margin. (https://days2011.scala-lang.org/sites/days2011/files/ws3-1-Hundt.pdf) The reason may be that “Go provides a model for software construction that makes dependency analysis easy and avoids much of the overhead of C-style include files and libraries.”

Good asynchronous primitives

What about Python and Java?

Python provides simple and powerful asynchronous frameworks (eventlet/gevent) to develop scalable backend services. In Java, you have to either go with threaded services like tomcat/jetty(new ones provides limited async support) or use reactor style libraries build on top of Java NIO like Netty. None of these come close to the expressiveness of python. Thus in this aspect Python would be a better substitute than Java.

Low level interfaces (manage processes, syscalls…)

What about Java and Python?

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression a+b, it must first inspect the objects a and b to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations for a and b, and does not allow overloading of the + operator. Therefore, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language.

Thus in this case Java would be a better alternative language than Python.

Extensive standard library and data types

What about Python and Java?

Both Python and Java have powerful and extensive standard library. Java’s STL is larger than Python’s. But both languages will be sufficient to handle the tasks.

Strong duck typing

One of the distinct features of Go is maximize the effect of duck typing. In Go, Interface is something like a “duck”, and the methods contained inside the interface structure is functionalities the “duck” can achieve. So in Go language, interface can be described as: duck with “this” and “that” functionalities.

What about Java and Python?

Python is famous for its duck typing. Thus in this aspect, Python would be a very good substitute. On the other hand, Java is a more rigorous languages, with less flexibility.

Full development environment

Both Java and Python can provide equally good or even better development environment than Go.

Multi-arc build

What is Multiarch?

Multiarch is the term being used to refer to the capability of a system to install and run applications of multiple different binary targets on the same system. For example running a i386-linux-gnu application on an amd64-linux-gnu system.

But, is Docker really making a good use of Multiarch?

Q: Docker is strongly tied to Linux by virtue of using Linux containers. So how does Docker benefit from the multi-arch build facilities of Go, if it only runs on Linux?

A: LXC does require Linux and x64 is the recommended architecture to run it on, however LXC can be built for other platforms (namely x86, x86-64, IA-64, PowerPC, SPARC, Itanium and ARM [1])

Q: But the Go compiler only has backends for three out of the seven architectures you listed. So this would seem to be a strike against Go.

A: …

Thus, it looks like Docker provides binaries for only x86-64 Ubuntu. So slide 24 seems to be more of "we think this is cool in theory" rather than "this really helps us."

Some final words about Dart and Go:

Both Dart and Go are developed by Google, but with quite different purposes.

The creation of Go was to solve the code engineering problems in large scale distributed computing, such as compilation and dependencies. It is focused on high concurrency environment. Thus Go is a server side static type language. During Go’s early days, it is described as “Internet’s C”. And I think Go will be very likely to reach that height someday in the future.

As for Dart, I want to first mention Javascript, a very popular language nowadays. Besides many of its cool stuff like Turn.js and AJAX, there is even a server side version of JS, which is Node.js. The purpose of Javascript is mainly for the Web front end; and that is the main purpose of Dart as well: to replace JS’s leading position in Web. After all, JS was developed more than a decade ago, with minor syntax changes over the years. It is hard to predict that if JS will be able to satisfy future needs.

AJAX: is a group of interrelated [Web development](http://en.wikipedia.org/wiki/Web_development) techniques used on the [client-side](http://en.wikipedia.org/wiki/Client-side) to create [asynchronous](http://en.wikipedia.org/wiki/Asynchronous_I/O) [Web applications](http://en.wikipedia.org/wiki/Web_application).

Turn.js: Turn.js is a JavaScript library that will make your content look like a real book or magazine using all the advantages of HTML5.

After roughly scanning through part of the Dart’s documentation’s for a while, I realize that Dart is aimed to reach a better language design and better maintenance than JS. The advantage of Dart may lies in the direction of Web components (**Web** Components is a set of specs which let **web** developers leverage their HTML, CSS and JavaScript knowledge to build widgets that can be reused easily and reliably.). It can be interpreted as a bridge between Web UI and data rendering.

Thus in short, both Go and Dart are languages with rather short history and pretty good potential. But they are designed to meet different needs and purposes. Go – server end. Dart – Web. Therefore Dart may not be a very good substitute to Go, but instead, it is a good complementary compared with Go.

As for choices between Java and Python. I think neither would be as suitable as Go in implementing Docker. Because Go is designed to merge the advantage of easy coding in Dynamic languages and advantage of security in static languages. So Java and Python will do some jobs as good as Go does. But neither of them will be as suitable as Go overall.