DockAlt: Alternative Implementations of Docker

Stanley Ku
University of California, Los Angeles
June 6, 2014

Abstract—The Docker platform provides lighweight, portable, self-containers as a method to pack, ship and run any software applications, The platform is implemented in Go, a relatively new programming language. This paper examines the feasiblility, both technically and politically, of other programming languages: Java, Python, and Dart as an alternative to Go in the Docker implementation.

Index Terms—Docker, LXC, Go, Java, Python, Dart

1 Introduction

The goal of Docker is to automate software deployment as lightweight, protable, self-sufficient container and builds on top of the Linux Container (LXC) concept. On a spectrum with virtual machines at one end and staticly linked binaries at the other, Docker is in the middle of the two. VMs abstracts an entire machine through virtualization and has a large overhead, thus limiting the number of runnable VMs on a single machine. Statically linked binaries assume that deployment environment has everything needed to execute the application; however that is not often the case. Docker builds on LXC, which enables sandboxing processes with namepsace and cgroups. The container concept allows developers worry about the application within the container and Ops worry about deploying the container.

2 Go

Docker started off as an internal project to rewrite parts of the dotCloud platform. The original implementation was in Python, but the developers wanted to start from a clean slate. The new language had to be neutral because opponents of the more common languages: Java, Python, Ruby, will limit customer adoption. Go fit this criteria because it was neutral and no bias against the language, at least at the time. For technical reasons, Go has some desirable features: static compilation, good asynchronous primitives, low-level interface, extensive libraries and data types and strong duck typing. Compile binaries have no dependencies and the ability to directly deploy the code to a server is a huge plus. There are a few drawnbacks being stemming from the young age of the Go language. There are still bugs in certain packages and the language itself that needs to be fixed. Go was chosen for both political and technical reasons.

2.1 Libcontainer

Docker introduced a new built-in execution driver, libcontainer, that ships alongside LXC. The libcontainer allows direct access to kernel's container APIs and allow manipulation to namespaces, control groups, network interfaces, etc. This eliminites dependency on LXC, which could vary between different versions and distributions. Therefore, LXC is actually now optional and one can switch between the two as one chooses.

2.2 Goroutines and Channels

Go is designed with asynchrony and concurrency in mind and not as an afterthought as shown by the gorountines and channel primitives. A goroutine is a function executing concurrently with other goroutines in the same address space. it is lightweight, costing little

more than the allocation of stack space. Initially the stack starts small so they're small and grow by allocating/deallocating heap storage as required. Goroutines might run in one or multiple threads but the implementation is abstracted. Channels are pipes that connect concurrent goroutines to allow for communciation.

3 JAVA

As an open source project, Docker relied on contributions from the community. Victor Vieux, one of the core engineer of Docker, quoted "I just completed the Tour of Go and felt competent enough to fix this issue, so be warned this [is] my first day with go ;)" in his presentation at GopherCon 2014. There is a very low barrier to entry for learning Go, which allows newcomer to learn the language and contribute. If DockAlt takes the same approach, this would not be possible with the overhead of learning a verbose and relatively more complicated language like Java.

On the same note, Java programs can become quite verbose which causes the code to become cryptic and difficult to understand. In addition, prototyping takes longer and leading to longer turnaround time.

3.1 Lack of LXC Support

Another downside of Java is that it doesn't support LXC binding. This would be a huge hurdle in implementing DockAlt. A possible solution is to use the Java Native Interface (JNI), which provides an escape hatch to call native libraries and code written in other languages to make sure of the LXC APIs. This causes the Java program to lose it's portability because it now depends on platform specific code.

4 PYTHON

Being the Docker precursor in the dotCloud framework was implemented in Python, Python is a tried and proven alternative. As a dynamically typed language, Python makes use of strong duck typing like Go to allow for greater flexibility. Python has LXC binding with the lxc package, which a huge benefit since Docker is built on top of LXC. Below are code to instantiate, create, start, stop and destroy a container in Python.

```
import lxc
container = lxc.Container('test_container')
container.create('busybox')
container.start()
container.wait("RUNNING", 5)
container.stop()
container.wait("STOPPED", 5)
container.destroy()
```

5 DART

Like Go, Dart is a language developed by Google. As such, it shares some of the same design methologies. Dart is primarily a web programming language that is an evolution of JavaScript and with the eventual goal of replacing JavaScript all together. It executes on DartVM, which can be integrated into the web broser or run as a standalone on a machine. Being devloped by Google, the DartVM is only currenly packaged with Google products like Chrome. Therefore, for compatability, Dart code is typically translated to JavaScript using dart2js compiler. This political issue conflicts with the neutrality requirement when initially choosing Go. Being even a newer language than Go, there will be bugs in the Dart language. If the goal for DockAlt is to be a safer alternative to Docker, then it doesn't make sense to choose Dart.

5.1 Lack of LXC Support

There is no LXC binding support in Dart, at least not that I'm aware of. There is LXC package for Node.js and since Dart evolved from JavaScript, I would predict that a LXC package will devloped for Dart/Dart equivalent of Node.js as the language gains more traction and matures. With the current lack of support for LXC binding, it is a significal disadvantage. On the bright side, it shouldn't be too much effort to develop one. I would assume that the LXC package for JavaScript/Node.js could be ported over.

5.2 Optional Typing

Typing in Dart is optional. This allows quick prototyping like in Python but with the added benefit of optionally adding typing to enable type checking and reduce runtime error. Like Go, Dart follows a strong asynchrony and concurrency paradigm. Dart has isolates, which similar to threads but with isolated memory heaps. The idea was that shared-memory threads are error prone. Isolates communicate through messages which introduce overhead. DockAlt could instantiate a container with each isolate. This provides reliability and robustness. Should a container crash, it will be confined to its isolate and would not crash the entire DockAlt engine.

6 Conclusion

Python is a strong choice for DockAlt because it is already tried and true in the dotCloud platform. Having LXC binding is huge benefit. Dart is a great alternative and the language introduces some great features that DockAlt could benefit from, but from it falls back from a reliability standpoint. As Dart matures, I think it will be surperior than Python for DockAlt but currently Python is the best choice.

References

[1] ”DOCKER 0.9: INTRODUCING EXECUTION DRIVERS AND LIBCONTAINER”, (Docker), http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/, (Accessed: 6 June 2014)

[2] Kurtovi, F.”LXC Python bindings”, http://sgros-students.blogspot.com/2013/04/lxc-python-bindings.html, (Accessed: 6 June 2014)