CS 131 Homework 6: Containerization support languages

Kaiwen Huang
University of California, Los Angeles

Abstract

This report compares three different languages, Java, Python and Scala in implementing an alternative rewrite of Docker, a type of Linux Container for operating system-level virtualization. The alternative rewrites are evaluated as opposed to the original langauge, Go, in which Docker is implemented.

1 Introduction

We want to avoid overhead involved in running large set of virtual machines when running our Twisted Places proxy servers. Using Linux Container is one solution to this problem. Specifically we are looking at Docker, a recently developed platform written in Go. Docker has some weaknesses such as it is new so maynot be very stable and it has single source which may be fragile to a fatal error. We want to investigate three other languages that may be potentially helpful in solving these weaknesses to implement DockAlt, a rewrite of Docker.

2 Containerization and Virutal Machine

Distributing software to different platforms becomes difficult when platform has their own libraries, functions and binary files. This problem is generally resolved by using virtual machine, which distribute a virtual disk that has applications along with its dependencies. Container uses a similar idea to virtual machine but with lightweight method. The main difference between a container and virtual machine lies in that virtual machine has a guest operating system running on top of the host operating system and this high-level abstraction consumes resources such as RAM and CPU; container does not have such a layer of abstraction but has only an "engine" that runs atop the host operating system. The container approach eliminates the overhead in simulating another kernel and the hardware. Docker is one type of such containers.

3 Docker in Go

Docker is implemented in Go, a relatively new programming language which is developed mainly for systems programming. Go is imperative and statically typed similar to C/C++. Go in short has two features: lightweight and good support for networking, concurrency and safety. Below are some categorized features of Go:

- Syntax: imperative language, close to C, simpler and more readable than C.
- Type: use type inference and has no type hierarchy; static typing; supports self-defined types
- Memory: has a primitive garbage collector in hybrid style: asynchronized and concurrent, mark and sweep.
- Concurrent programming: have built-in primitives as goroutines and channels to support parallel programming
- Linking and loading: Statically linked native binaries without external dependencies; has remote package management system with hierarchical package naming
- Compilation: statically compiled, fast

The above features make Go a very nice langauage to work with, primarily because of its simple syntax, good type and memory safety, asynchronous primitives and platform independence. However, Go is still a young language with a relatively small community. Its tooling has limitations and its error handling is usually difficult.

4 Alternative Docker: DockAlt

This section discusses three other languages as candidates for implementing a rewrite of Docker, DockAlt.

4.1 DockAlt in Java

Java is an imperative and object-oriented language. Java source code is compiled into byte code and then runs on a Java Virtual Machine (JVM). Java is strongly and statically typed with type hierarchy. Garbage collectors in Java depends on the implementation of JVM, some of which have concurrent mark and sweep garbage collecting. The native libaries in Java supports a variety of requirements. For implementing DockAlt, Java has the following advantages/disadvantages compared to the other languages:
Advantages:

- Safety and Reliability: Java's strongly typed system ensures good safety via compile time type checking. Type errors can be caught at compile time and prevent possible crashes when program runs. Java implements explicit exceptions in type. It is ensured that caller function handles the exceptions thrown by callee functions, which reinforce the safety in types related issues. Garbage collector also ensures safety in memory.
- Paltform Independence: Java code can run on any platforms because all platforms can run a standard JVM. This eliminates the need for different versions of DockAlt source code. A single version is sufficient for all platforms. Go also has a similar advantage of platform agnosticity.
- Concurrent programming: Java has lightweight libraries for parallel programming, and in this sense it is better than Python.
- Availability: Java is a very popular language with good community support and its libraries and packages provide developers with reliable source code that can implement a variety of funcitons and requirements.

Disadvantages:

- Compile time: the complexity of Java's type system makes the compilation slower than Go, which has fast compilation due to the lack of type hierarchy.
- Ease of use: Java is heavy weight compare to Go and Python. It may be hard to use from scratch and the syntax may take time to understand. Because it has no type inference, declaring variables makes the code lengthy. Java has longer development cycle, which is not good for fast prototyping as needed by Docker.
- Support for LXC: Java does not have native LXC interface that Docker is built on. This makes usage of LXC in developing DockAlt more difficult and thus may slow down the development cycle of DockAlt.

4.2 DockAlt in Python

Python is an imperative, object oriented scripting language and it is interpreted by interpreter. Python byte code is generated from Python source code and then it is interpreted or just-in-time compiled. Python is strongly and dynamically typed with duck typing. Garbage collection in Standard Python implementation uses reference counting. For implementing DockAlt, Python has the following advantages/disadvantages compared to the other languages:
Advantages:

- Ease of use: Python is known for its simplicity in syntax, which makes it easy for developers to read and use. This also can make prototyping and debugging in DockAlt easier.
- Package Management: unlike Java, Python has native support for LXC which makes developing Docker faster and easier. Also Python has many packages for various purpose and allows customizing the versions of the packages. This is better than Go because the Docker developers complain about "go get" can not pin a particular version.

Disadvantages:

- Performance: Python is interpreted language so it runs slower than Java and Go. Since Docker is targeted as a lightweight platform that can be used by many applications, efficiency may be more important than simplicity in code.
- Safety and Reliability: the dynamic type checking in Python defer the catches on type errors to runtime, which may cause danger if error go unnoticed. This may be fatal for large-scale distributed software like DockAlt.

4.3 DockAlt in Scala

Scala is a language with both objected-oriented and functional features. For objected-oriented part, each value in Scala is an object and each operation is a

method call. For functional part, functions in Scala are treated as objects and function type is a class. For implementing DockAlt, Scala has the following advantages/disadvantages compared to the other languages:
Advantages:

- Simplicity: Scala has concise syntax and it is possible to use a single word to achieve several loops. The lightweight in syntax is comparable with those in Python and Go.
- Safety and Reliability: Similar to Java, Scala is strongly and statically typed and errors can be caught at compile time. In large code bases, this ensures more safety thus is more desirable than runtime typechecking.
- Performance: Libraries of Scala have tuned and optimized collection operations, which makes collection usage efficient. Collections can also be adapted to parallel execution on multi-cores. This is a plus for the parallel programming in DockAlt implementation.
- Concurrent Programming: Scala standard library has parallel collections to facilitate parallel programming. It also supports concurrency through many other built-in classes and third-party framworks.
- Flexibility: Scala can create objects, call methods in Java. It can use Java classes and implement Java interfaces freely. This compatibility with Java introduces more options in tools to choose from than native libraries. Also Scala can run on JVM and other java-based platforms such as Elipse, this allows compiled Scala packages to be used in Java development as well.
- Platform Independence: Scala byte code can be run on JVM, making Scala platform agnostic.

Disadvantages:

- Ease of use: Scala is a strongly functional language although it does support imperative-style implementation. Scala is hard to use in the sense that it has a totally different programming paradigm from traditional languages such as Java and it has high requirement on developers' skill set to master it.
- Community Support: Scala is new so there are limitations in finding solutions and help from communities. This may slow down the development cycle of DockAlt if the developers have to resolve issues by themselves. Go as a new langauge also suffers similar problem.
- Backward Compatibility: New releases of Scala are mostly incompatible with old versions. This may bring a lot of inconvenience for developers of DockAlt since they have to fix the differences in the code each time the langauge is upgraded to a new version. Java is better than Scala in this sense because Java versions are mostly binary backwards-compatible.

5 Conclusion

Based on the research on the three alternative languages for DockAlt, we can see that Scala shares with Go the advantages of having good lightweight frameworks to support concurrency albeit through different ways. However they are both very new languages that lack sufficient community resources. Go may be eaiser to learn and to understand than Scala. Java is a well-developed langauge with good community support but it does not have native LXC support and its syntax and type system adds complexity for developing the Docker. Python seems a better choice than the other three because it is a mature langauge that is easy to start with and have large community support; it has automatic memory management and sufficient libraries and packages. Most importantly it has native LXC interfaces which are especially good for developing Docker.

References

[1] Tobby Banerjee, "Understanding the key differences between LXC and Docker": https://www.flockport.com/lxc-vs-docker/

[2] Docker documentation: https://www.docker.com/

[3] Linux container documentation: https://linuxcontainers.org/

[4] Michael Crosby, Jerome Petazzoni, Victor Vieux, Docker: an insider view, http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/22-Why_Go3_it_has_what

[5] Golang: pros and cons: https://altabel.wordpress.com/2015/11/10/golang-pros-and-cons/

[6] Java Pros And Cons: http://c2.com/cgi/wiki?JavaProsAndCons

[7] Paul Krill, "A developer's guide to the pros and cons of Python":

http://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html

[8] Scala documentation: http://docs.scala-lang.org/overviews/collections/introduction.html

[9] Harry Ulrich, "Pros & Cons of Scala Programming for Agile Software Development": http://blog.celerity.com/prosconsscala