Containerization Support Languages

Jinjing Zhou
University of California – Los Angeles

Abstract

Containers, an OS-level virtualization of the "user space", is a concept that has been introduced to help with various development since the early 2000s and it has been made available by Docker. Advantages of containers include that it is extremely lightweight, has very low overhead, allows predictable and reproducible behavior, and ensures modularity and scalability. However, due to the fact that it does not have a separate operating system/kernel, it is required to be less isolated from the kernel, which should be taken into consideration when choosing the programming language for implementing Docker. In addition, the language should also be able to be fast, able to do asynchronous programming as well as other traits that will be further demonstrated in this report.

1. Go

The language used for the current Docker implementation is Go. Go is a programming language created at Google in 2009. It has traits such as static compilation, static typing (with limited structural typing), memory safety features and communicating sequential processes –style concurrent programming features.[1] It was built with concurrency in mind, and has goroutines instead of threads, which can have millions of running at the same time, making it free of deadlocks but with even faster startup and running speeds.

1.1. Advantage

There are various reasons why the developers chose Go to implement Docker. First of all, Go uses static compilation and static typing. Once you run the command "go build", it will install everything needed in order to run the program. This will make programs written in this language fast and easy to execute, test and adopt. In addition, it has good asynchronous primitives, thus providing the kind of running speed that Docker needs. It also low-level interfaces, extensive standard libraries and data types, which allows Docker to work closely with the kernel and provides faster system calls. Furthermore, it provides suitable development environment for Docker with its functions such as "go doc" and "go get" which provides documentation for packages and fetches dependencies on github.

1.2. Disadvantage

However, Go has its drawbacks. Besides not being as mature as other programming languages such as Java and Ocaml, it is fast at the expense of not being thread-safe, and it has no mechanism like mutexes to protect accesses. Also, its error-handling can be verbose, which can cause much trouble in the development process. Last but not least, though goroutines can be much faster than threads, they cannot select on readers and writers, so everything has to be reduced to channels, increasing the level of difficulty during the development process.

2. OCaml

OCaml is one of the ML-derived languages that are known for their static type systems and type-inferring compilers. OCaml, in particular, combines functional, imperative and object-oriented programming under an ML-like type system.

2.1. Advantage

As OCaml is strongly and statically typed and has the most powerful type inferring, all errors will be detected at compile time, simplifying the development process to a large extent.[2] Similar to Go, OCaml also uses static compilation, so all source codes are compiled down to bytecode. This high portability and characteristic of being machine agnostic is exactly what is required by DockAlt. In addition, it also provides powerful libraries and low-level tools, making it a good choice for DockAlt by allowing DockAlt to use the users' kernel easily. Another advantage that makes OCaml an even better choice than Go would be its powerful integrate macro libraries, which lets programmers to alter its syntax on-the-fly. As a result, it is much easier and quicker to write parsers, thereby do error-handling, which can be rather crucial in the development process of DockAlt.

2.2. Disadvantage

While OCaml has the library Async that allows it to handle asynchronous programming, which is required in implementing event-driven servers well, the library Async takes the multi-threaded approach in achieving this and most mutable OCaml data structures do not have well-defined semantics when accessed concurrently by multiple threads. Therefore, if DockAlt is implemented with OCaml, it has to implement mechanisms such as mutexes to avoid race conditions and achieve thread-safety, which will slow down the code even with Nano_mutex, a faster alternative than OS-level mutexes.[3] In addition, OCaml's poor multicore support may lead to poor performance in today's computers, which is quite the opposite from Go.

3. Java

Java is a general-purpose programming language that is concurrent, class-based, object-oriented, and specifically designed to have the least amount of implementation dependencies possible. The primary goals upon the creation of Java include being robust, secure, threaded, interpreted, and dynamic, as well as with high performance.

3.1. Advantage

Like Go and OCaml mentioned before, Java also has static compilation, which means all Java codes will be compiled into bytecode, so that it is runnable on Java virtual machine on any computer architecture, making it great for multi-platform applications, which, in our case, is DockAlt. As one of the most popular languages, Java is equipped with not only a large and standard class library, but also lots of available codes and third-party libraries. Other advantages of Java include short learning curves due to simplified syntax, comprehensive documentation and automatic memory management[4], which will all significantly speed up the development process.

3.2. Disadvantage

Since Java supports asynchronous programming through multi-threading, it shares the same problem as other programming languages that support multi-threading, that upon concurrent execution, problems such as race conditions and deadlocks may happen. In addition, creating new threads in Java is not memory efficient as every thread consumes approximately 1MB of the memory heap size and will eventually put tremendous pressure on the heap if there are too many threads running, which may result in shutdown due to out of memory. It is also hard to do inter-thread communication. Furthermore, Java virtual machine is not necessarily on every machine, so in order to have DockAlt feasible on all platforms, the easiest solution would be have Java virtual machine in DockAlt, which will make the package larger and take up unnecessary space as well as make the installation process more complicated. Besides, Java is considered as a high-level language since Java source codes are compiled into machine codes that are understood by Java virtual machine, which then gives instruction to system, making Java have strong abstraction from the computer. This makes it more difficult for Java to make system calls and work closely with the user's kernel if it used to build DockAlt compared other low-level languages.

4. Scala

Scala is a functional and object-oriented programming language that runs on Java virtual machine, so Scala and Java can be freely mixed for seamless integration. It has strong type inference, supports concurrency and asynchronous programming.

4.1. Advantage

The static typing and strong type inference allows quick error-catching, just like OCaml. Unlike Java, on which it is based, since Scala has inherently immutable objects, it reduces thread-safety concerns that is a significant disadvantage in Java. In addition, Scala provides powerful concurrency through its reactive core and asynchronous libraries[5], and it has an Actor library that helps solve concurrency problems more rapidly. Together, they guarantee true multithreading and the ability to aggregate multiple databases.

4.2. Disadvantage

Since Scala has a very sophisticated compiler, which runs over 25 phases during compilation, the compile time for Scala may be an issue considering being quick is essential to DockAlt. Also, each new version of Scala

tends to be incompatible with the previous version, and this can cause much pain during a development process for an application using Scala. Last but not least, the arcane syntax and the though improving but buggy IDE of Scala may result in issues with learning curve.

5. Conclusion

While OCaml, Java and Scala all have their advantages and disadvantages as the language of implementation for DockAlt, I would choose OCaml as the language to implement DockAlt. Comparing all the three languages listed above, like the other two, OCaml is able to do asynchronous programming so it supports event-driven servers well, but it has faster compile time than Scala while the same (or even stronger) type checking power. Being a low-level language, OCaml will be able to make faster system calls, and it is also more portable than Java. It is also a more mature language compared to Scala, also a functional language, ensuring the development process for DockAlt to be more stable and predictable. In addition, it even allows for easy error-handling and solves the issue with Go. In conclusion, I feel that OCaml would be the best choice for implementing DockAlt.

6. Citations

1 Cade Metz 5 May 2011 at 19:18 tweet_btn(). "Google Go boldly goes where no code has gone before." The Register® - Biting the hand that feeds IT. Accessed December 06, 2017. https://www.theregister.co.uk/2011/05/05/google_go/.

2 "Strong Static Typing with Type Inference." OCaml for the Skeptical: Strong Static Typing with Type Inference. Accessed December 06, 2017. http://www2.lib.uchicago.edu/keith/ocaml-class/static.html.

3 Chapter 18, Hickey, Jason, Anil Madhavapeddy, and Yaron Minsky. Real World OCaml. Sebastopol, CA: OReilly, 2014.

4 "Java: Pros And Cons." Wiki.c2.com. Accessed December 06, 2017. http://wiki.c2.com/?JavaProsAndCons.

5 "Why we love Scala at Coursera." Building Coursera | Why we love Scala at Coursera. Accessed December 07, 2017. https://building.coursera.org/blog/2014/02/18/why-we-love-scala-at-coursera/.