Artiom Arutiunov
UID: 504597668
Section 1D

Homework 6 Report

Introduction

The purpose of this assessment is to provide an introduction to the functionality of Docker as well as assess the benefits and costs of Java, OCaml, and Rust as replacements for Go to write an alternative version of Docker.

About Docker

What is Docker?

According to the presentation referred to in the project specification, at a high level, Docker is an application container, and at a low level, a machine container[9]. In terms of being an application container, Docker emulates normal processes of the program- they are just isolated from the rest of the machine. Moreover, Docker shares the kernel with the host and does not emulate the device it is running. In terms of being a machine container, Docker creates its own process space, network interface, and run stuff as its "root." In general, Docker is a way to package everything an application needs into a single container, including any libraries the application may be using. On the developer side, Docker makes it easier to identify problems and inconsistencies within the code via its isolation from the machine it is running on; conversely, on the client side, Docker makes it easier for clients to run the application because they do not have to actually install it onto their machine.

Why Go?

Docker is written in the Go programming language. According to the same presentation, the reasons for using Go as the language of choice are fivefold[9]. The first reason is that Go is a static programming language. Static compilation is safer because the compiler knows exactly what libraries are needed in the application at compile time, and can ensure the entire application works before it is executed. The second reason is that it is a "neutral" programming language, meaning it is, in the words of the presenters, "not C++…Python…Ruby…[or] not Java[1]." It is a language that does not adopt any of the entirely unique characteristics of the four aforementioned programming languages, which can cause intrinsic issues in terms of the functionality of Docker. The third reason is because Go has features that are needed for writing an independent container such as Docker, including asynchronous primitives (which is needed for I/O, waiting for processes, etc.), low-level interfaces (such as managing syscalls), extensive standard library and data types, and, lastly, strong duck typing. Duck typing has to do with type safety - specifically, that type checking be deferred to runtime, and that an object's sustainability is determined by its properties and methods rather than by the type of object itself. The fourth reason is that it addresses multiple issues of the development workflow. Finally, the last reason is that Go has a multi-arch build.

Disadvantages

One of the key disadvantages of Go is that it is not thread-safe[9]. Namely, maps in Go aren't thread-safe, despite being fast. It is on the developer to make sure no threads step onto each other's toes and that there are no race conditions in the code. Moreover, the "get", "test", and "build" commands have their inconsistencies and inconveniences that cannot be overlooked. Go also does not have a standard IDE, meaning that standard text editors like emacs or vim have to used. However, as mentioned earlier, Go's trait of static compilation can be a serious drawback in terms of development. If one changes even one tiny facet of the application or libraries it uses, the entire application needs to be re-compiled. For large projects, this can take a large amount of time and can prove to be very time-consuming. According to IBM, statically-compiled executables are as a result larger files[11]. Lastly, a drawback of Go is that it can print verbose error messages[9].

Java

What is Java?

According to IBM's documentation on Java[11], Java is an object-oriented, dynamically compiled language. It is one of the most widely used languages in the world.

Advantages

There are a few noteworthy characteristics of Java that would prove useful in using it to develop an alternate version of Docker. First, according to the official website of Java, it has static type checking, meaning that many errors and bugs can be caught and avoided during compile time instead of runtime[3]. Moreover, Java is multithreaded[8]. This allows for multitasking, and multithreaded programming has been fluidly integrated in Java. Java is also highly portable because it is platform-independent and distributed via byte code instead of machine code. Another salient positive of Java is that there has been a huge amount of pre-written libraries that can be used in development.

Disadvantages

According to Rafael Benevides of the Red Hat Developer Program, a notable disadvantage of Java is that it has no default binding to Linux container[1]. Thus, it is much more difficult to write low-level code in Java without incorporating (often hefty) external libraries. This could make the overall application larger than originally intended. Moreover, as the official Java website points out, the Java Virtual Machine (JVM) is not by default found on every computer[3]. Therefore, one would have to incorporate JVM with the Docker package, making the overall product also larger than originally intended. Thus, in terms of the ability to write low-level and data-efficient code, Java is at a disadvantage.

Ocaml

What is Ocaml?

According to its official site, Ocaml is "a general purpose programming language with an emphasis on expressiveness and safety.[10]" In general Ocaml can be utilized in the object-oriented or functional programming style.

Advantages

The official website of Ocaml lays out a few of its most prominent advantages[10]. For one, its brevity and speed in code are some of its standout advantages. Specifically, Ocaml features a "powerful type system", where type inference is handled at compile time. In addition, according to Chris Donaher of Endgame, Ocaml has a plethora of powerful libraries that make writing low-level code simple

efficient[5]. Like Java, Ocaml is very portable because it compiled down to bytecode. Lastly, Ocaml does not store local variable in memory; instead, it "curries" functions, meaning that is passes functions as argument to other functions. This makes Ocaml code on the whole lightweight, fast, and easy to understand.

Disadvantages

According to the official Ocaml website's documentation[10], Ocaml does not support multithreading. It does support concurrency however, but it is therefore a greater burden on the developer to write code that could be concurrently executed instead of multithreaded. Moreover, Ocaml does heavily rely on its functional nature, which is drastically different than the object-oriented approach of languages such as C++ or Java. It can take developers time to really understand how to think in a functional fashion, which can be quite challenging when implementing low-level code for the Docker. Lastly, as Benedikt Meurer points out in his stud of Ocaml byte-code[7], Ocaml requires a just-in-time compiler (JIT), which is a similar dilemma to the issue of a JVM needed for Java. Including a JIT in the Docker package would increase the overall size of the product, or would require the client to already have a JIT installed prior to using Docker.

Rust

What is Rust?

According to Rust's official website, it is a systems programming language that is safe, concurrent, and practical[6]. Its syntactic style is similar to that of C++, and its overarching design is to ensure memory safety while maintaining performance.

Advantages

The biggest attraction of Rust is that it is designed to be highly concurrent and highly memory-safe. As Charlie Crawford writes in TheNewStack, Rust is a "low-level language, best suited for systems, embedded, and other performance critical code.[2]" This is especially beneficial for developers: The simultaneous ability to write low-level and safe code ensures that development with Rust will create a functional product. Moreover, Rust's syntax is similar to that of C++, and one can take advantage of programming paradigms similar to object-oriented, which is familiar to a wide breadth of developers. Moreover, Rust gives developers the ability to write unsafe, yet faster-performing code - though Rust defaults to safe code in its functionality. A final benefit to Rust is that it serves as a great introduction to systems-level programming. Rather than learning a language such as C which lacks many of the defining features of an object-oriented language, developers can use their prior knowledge via other languages and behind to engineer the Docker product.

Disadvantages

Rust is an open source programming language. As a result, there isn't a centralized body that governs the development of the language, which could make it more difficult to maintain the Docker product via Rust. For example, updates or patches would need to be made based off new development the open source body decides to implement. Furthermore, Rust is a very syntactically explicit language. For example, provides the developer with a much wider range of choices to use for types. This can result in developers not even recognizing that they are using a wrong type, method, or library due to such a wide array of design choices.

Conclusion

Based on my research, I conclude that Rust is the best alternative to Go to write a new version of Docker. For one, it closely resembles languages such as C++ in its object oriented design. In addition, it is very adept at writing low-level code. It can also be parallelized, and has a growing community of developers that increase its scope, power, and functionality on a daily basis. According to StackOverflow, Rust came in first place in terms of the most "beloved programming language" among developers. Despite its drawbacks, its recent birth yet spanning popularity attests to its effectiveness as a programming language.

Bibliography

1) Benevides, Rafael. "Java inside docker: What you must know to not FAIL." RedHat, 14 Mar. 2017, developers.redhat.com/blog/2017/03/14/java-inside-docker/.

2) Crawford, Charlie. "How Rust Compares to Other Programming Languages." The New Stack, 2 May 2016, thenewstack.io/safer-future-rust/.

3) "Develop Software." GoJava, go.java/developer-opportunities/index.html.

4) Disadvantages of Java. ACM, 1997, www2.gsu.edu/~matknk/java/reg97-5.htm.

5) Donaher, Chris. "Build Safer Programs Faster With Ocaml." Endgame., 24 June 2016, www.endgame.com/blog/technical-blog/build-safer-programs-faster-ocaml.

6) "Introduction." Rust, doc.rust-lang.org/book/second-edition/.

7) Meurer, Benedikt. Just-In-Time compilation of OCaml byte-Code. Universität Siegen, 27 Sept. 2011, arxiv.org/pdf/1011.6223.pdf.

8) Patel, Kamlesh. "Most Significant Advantages of Java Language." Streetdirectory.com, www.streetdirectory.com/travel_guide/114362/programming/most_significant_advantages_of_java_language.html.

9) Petazzoni, Jérôme. "Docker and Go: why did we decide to write Docker in Go?" LinkedIn, 7 Nov. 2013, www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go/32-no_IDELets_ask_Samuel_Jacksonwhat.

10) "What is Ocaml?" Ocaml.org, ocaml.org/learn/description.html.

11) "When to use dynamic linking and static linking." IBM Knowledge Center, IBM, www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.performance/when_dyn_linking_static_linking.htm.