An Alternative to Go in Docker: DockerAlt

***Abstract**: Docker is a software deployment container that focuses on allowing developers to deploy their applications in a lightweight, portable way. Docker was implemented using Go, a programming language devised by Google. This paper examines alternative programming language choices from Java, OCaml, and Dart.*

Introduction to Docker

Docker is a product that deploys software inside of a software container to ease deployment with portability and self-sufficiency. It has advantages over virtual machines which have been the traditional method or software deployment since Docker has less overhead than a virtual machine while still allowing the sandboxing of an application. It abstracts away the trouble of deployment so that developers can worry less about software application portability.

Design Choices in Docker's Development

The creators of Docker wanted

1. static type checking
2. good asynchronous primitives
3. low level interfaces
4. extensive standard library
5. duck typing
6. ease of development

Go

Go was the programming language of choice chosen by the developers of Docker. While making Docker, they first specifically paid attention to selecting and language which allowed for static compilation. Go is a statically typed language which also supports type inference. The creators of Docker also wanted a language which had good support for asynchronous primitives. Go was built for concurrency and asynchronous behavior to allow for great performance. A low level interface and extensive built in standard library were also important in the creation of Docker. Golang met these needs as well since the language itself has a lot builtin to it. Go also made the development experience nice too which allowed developers to write less code and to easily and quickly build their application.

But a problem with Go is that, as a relatively new language, it is still not stable yet and mature external libraries and support do not exist yet for it.

Java

From the point of view of what is best for Docker, Java is nice in that it has a strong static typing. Java also has built in garbage collection like Go. Java's JVM is meant to be portable so that aligns with the goal of Docker — to make software applications portable. Java also has asynchronous primitives which is what the developers of Docker were looking for. Java has a rich standard library. Another plus of using Java is that it's a mature and popular language. Thus finding developers is easy and developer ramp up time on a Java project would be much easier than if all new developers were new to a less mature language.

On the other hand, Java is not functional in nature. Thus we would be missing out on the great functional language features which are efficient and lead to more elegant solutions. Java also has some controversy over having security issues in web-based applications. Java is also slower than C++ and Go because of the overhead of its JVM and garbage collection. Java is also a very verbose and complex language which leads the code to be harder to understand.

OCaml

OCaml could be a good language of choice because it runs pretty fast. It also has static type checking with type inference. Thus this is why OCaml programs are faster, because there is no need for runtime type checking after compilation. OCaml is portable across varying operating systems and CPUs. OCaml also has pattern matching as a language feature which is quite useful during development.

Some downsides of using OCaml are that OCaml is not a mainstream language thus there is less developer support and open sourced external libraries that are still frequently updated and maintained. It also has no primitives for parallelism which is a huge downside for Docker's needs. OCaml also has incomplete support for debugging since it's error messages can be quite cryptic at times. Thus the developer experience is not as comfortable and may take longer to get the same amount done.

Dart

Dart is a relatively new language also maintained by Google like how Go is. Dart is dynamically typed with type inference with a static type checker that shows warnings if a type error is detected. So although the language is dynamically typed, it warns the developer of errors early on. Dart also allows for optional type annotations which means the developer can explicitly hint at the compiler that this object is of some specific type. This mimics static type checking. Thus Dart's approach to type checking allows the great flexibility of dynamic type checking while still maintaining the safety of static typing. Developers can also run a Dart program in checked mode to find any typing errors.

Dart has an extensive standard library with support for asynchronous programming. It runs on the Dart Virtual Machine with the goal of being used to build web and server applications.

A negative of using Dart is that the language is in fact dynamically typed thus is not as safe as a statically typed language. Additionally, Dart does not have much support for low level interfaces.

Dart is also again not a mainstream language so finding developers for it will be harder and the overall development ramp up will take longer.

Conclusion

If we were to develop Docker, Dart seems like a good option since it allows for flexibility in development while still supporting static type inference. It supports asynchronous programming and has potential as it matures to become a good choice like Go. Dart is not as good as Go yet but with time it may become equal. Go was a good choice to develop Docker since it was designed for scalability, concurrency, and parallelism, and I see Dart going in that same direction.