CS 131 Homework 6: Containerization support languages

Abstract:
Docker is a lightweight open platform that allows users to build, ship, and run applications on any computer, on any infrastructure, and in the cloud. Docker is implemented in the Go programming language and uses operating system level virtualization with Linux containers (LXC) . This paper will examine feasibility of implementing a separate implementation of Docker (DockAlt) in Java, OCaml, and Hack.

1. Introduction

Docker builds on LXC to create a sandbox environment. Applications need necessary binaries and libraries, language-specific tooling, and the setup and maintenance of development environments across the different machines developers are using. Docker allows developers to not worry about this problem, avoiding "dependency hell"[2]. Docker containers include the application and all of its dependencies while sharing the kernel with other containers. These containers run as isolated processes and are not tied to any infrastructure. Furthermore, Docker containers enable developers to use the language that best solves the problem at hand as the isolation of containers means internal tooling conflicts isn't a problem [1]. However, due to the relatively new nature of Docker, we want to implement a backup in another language in case the main one is buggy. We will discuss the pros and cons of writing the Docker alternative in Java, Ocaml, and Hack.

2. Docker in Go

One of the reasons Docker was written in Go was due to the fact that Go is a neutral language in hopes of sparking less controversy. Let's take a look at some of the pros and cons of using Go so that we have a basis of which to compare the alternative languages to [3].

Pros:

- fast static compilation
- easy dependency analysis
- no hierarchy (save time by not defining type relationships)
- static types (made to feel lightweight compared to typical OO languages)
- fully garbage-collected
- support for concurrent as well as asynchronous execution and communication
- Has an approach for construction of system software on multi-core machines

Cons:

- map's aren't thread-safe
  - but they are fast
  - can protect access with sync.Mutex or channels of channels
- can't pin a particular revision so source code of vendors had to be imported to the Docker repo
- can't have destructors and cleanups in tests
- painful to build multiple binaries if they share some common code

3. Docker in Java

Java is a very popular language that has a lot of support and libraries for it. Thus it is generally pretty reliable and good for writing large apps such as this one. However we would be picking a non neutral language so some people may not like this. Let's look at more technical details of the language [4 & 5].

Pros:

- Object Oriented
- Static typing
  - good for catching errors and debugging, especially so for larger programs such as this
  - results in slower compilation than Go

- Readability
- Memory Management & Garbage Collection
    - Uses a heap and the nursery and generation method
    - Uses mark and sweep for garbage collection
- Good performance and support for concurrent programming
- Easy to port to multiple platforms

Cons:

- Poor support for functional programming
- Can take longer to code
- No LXC bindings
    - This is very unfortunate as LXC is a core part of our approach which would add a lot of development time

4. Docker in OCaml

OCaml is a powerful language that is gaining popularity and is accepted by a good amount of programmers so it we can say that it is a pretty neutral language [7-9].

Pros:

- Object oriented, functional, and imperative programming support
    - lots of tools to aid in writing code
- Strong static typing and type inference
    - Good debugging and error catching
- Pattern Matching
- Memory Management & Garbage Collection:
    - Generations memory management
    - Stop&Copy does minor garbage collection on new generations
    - Mark & Sweep does garbage collection on old generations
- Efficient compilation

Cons:

- No LXC binding support
- No overloading
    - can make writing code more tedious
- Poor multi-core support
    - Although this is changing as people are looking into making OCaml multi-core safe and companies like Jane Street have successfully used parallel processing with message passing with good performance and security
    - This means lower performance in cases where we need to use more than one core
- No overriding equality, comparison, and hashing for new types and no way to catch errors that are a result of this
    - Type classes and equality types could help solve this but this just makes code harder to write
- No value types results in more memory being used and reduced performance
- No floats and no try..finally construct among some other missing functionality

5. Docker in Hack

Hack is a relatively new programming language that was developed by Facebook. This does mean that is has some good support due to corporate backing. However it has been open-sourced, thus it is a bit murky if this is a neutral language .

Pros:

- "Gradually typed"
    - dynamically typed code that interoperates with statically typed code [10]
- Instantaneous and quick type checking
- Asynchronous programming support
- Tracing garbage collection
    - Traces from a root set and build a reachable set. Any values on the heap that are unreachable are assumed to be free

Cons:

- Primarily intended for front end web development
- No LXC binding support

6. Conclusion

The goal is to write an alternative for Docker so the ideals should remain mostly aligned with it unless there are areas where compromising these ideals would give benefits such as performance, readability, ease of debugging, and so on. This is not going to be a front end project, thus Hack is probably not the best choice and it won't give us the amount of control we are looking for. Hack also doesn't have as much support for it compared to the other languages. This brings the discussion down to Java and OCaml. Neutrality wise OCaml takes a win over Java. OCaml has good debugging and error catching which is a big plus, although Java is good too. Java seems to be safer when going the multithread route but, OCaml has other options such as parallel processing and message passing. Furthermore, a multi-core OCaml is under the works and OCaml. So OCaml still looks promising and it is very safe. OCaml also lets us use functional programming and has a good amount of libraries. Furthermore the High-Level Virtual Machine was coded in OCaml in around 2,000 lines of code [14]. This is a good example that OCaml works for problems related to what we are trying to do and there is a good chance it will work on our project as well. Furthermore, Java can be very verbose and OCaml would help reduce code size and can possible even increase readability. Therefore I defiantly think writing DockAlt in OCaml would be worth trying out over Hack and Java.

7. References

1. What is Docker?, https://www.docker.com/what-docker#/Advantage
2. Docker Github, https://github.com/docker/docker
3. Docker and Go: why did we decide to write Docker in Go, http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go
4. Java pros and cons, http://luke.breuer.com/time/item/Java_pros_and_cons/185.aspx
5. Understanding, Memory Management, https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html
6. Memory Management by Objective CAML, https://caml.inria.fr/pub/docs/oreilly-book/html/book-ora087.html
7. What is OCaml?, https://ocaml.org/learn/description.html
8. Why OCaml?, http://www2.lib.uchicago.edu/keith/ocaml-class/why.html
9. Pros and cons of OCaml, http://flyingfrogblog.blogspot.com/2011/03/pros-and-cons-of-ocaml.html
10. Hack: A new programming language for HHVM, https://code.facebook.com/posts/264544830379293/hack-a-new-programming-language-for-hhvm/
11. Hack Documentation, https://docs.hhvm.com/hack/
12. On Garbage Collection, http://hhvm.com/blog/431/on-garbage-collection
13. OCaml and Multithreading, http://ib-krajewski.blogspot.com/2015/11/ocaml-and-multithreading.html
14. What is the best programming language for coding a virtual machine and or virtual CPU?, https://www.quora.com/What-is-the-best-programming-language-for-coding-a-virtual-machine-and-or-virtual-CPU