Homework 6: Evaluate A Language For A Garage-sale Buyer App

Name
University of California, Los Angeles

Abstract

There are many programming languages out there. When we want to develop a mobile application with a heavy usage of CPU-bound operations (e.g. running machine learning algorithms), we can use Dart for the development. But there are also many other options such as OCaml, Java, and Python. When we compare the features of these programming languages in terms of compilation, multi-threading, and garbage collection, it turns out Dart is the best language to use among these options for the type of application mentioned.

1. Introduction

We want to build a garage-sale buyer app, where users can walk into a garage sale, take a panorama of the items and price tags, then the app tells them what the best deals are. One way to do this is to send the image to the server, and let the server do the processing. Finally, receive the response from the server. However, this takes too much network bandwidth (we don't want to consume too much of users' monthly available data). To fix this problem, we can take advantage of AI accelerators on modern phones. We can use TensorFlow Lite for running the algorithm on the client-side. Now, we want to investigate whether Flutter user interface toolkit is a good technology to use for developing such an application. Let's compare Dart, the programming language for developing Flutter apps, with other language alternatives such as OCaml, Java, and Python.

2. OCaml

Ocaml is a statically typed functional programming language. It has type inferences so type annotation is not required. OCaml is compiled to machine code, but it also provides top-level interactive loop like an interpreter.

OCaml uses generational garbage collection, which is discussed in great detail in the Java section.

At the moment of the writing, OCaml supports threading with Global Interpreter Lock (GIL), which is described in great detail in the Python section. GIL essentially prevents the program from being executed in parallel. OCaml is currently working on this feature to support shared-memory parallelism.

3. Java

Java is a statically typed object oriented programming language. Java gained its popularity due to its portability. A Java program compiled to bytecode will work on any platform with Java Virtual Machine (JVM). Thus developers need not worry about the different architecture of the machines running the program. There isn't a good library/framework for cross-platform application in Java, but we will assume there is one, and focus on Java's features as a programming language.

Java has multithreading, which allows code to be run in parallel by placing threads on different cores of the machine, and thus, maximize the utilization of CPU.

Java used to run plain garbage collection with mark-and-sweep algorithm as an Daemon thread (i.e. a low priority thread). This algorithm has two phases. In the mark phase, it marks all the objects that are reachable from the root. Note that this must be done recursively, so objects that are reachable from objects that are reachable from the root are also reachable from the root (i.e. R is reachable to A and A is reachable to B, then R is reachable to B). This is not efficient because the time complexity of marking and sweeping grows as the number of objects in the program grows. Java now uses a more efficient algorithm called generational garbage collection, which uses the nature that newly created objects likely become unused and long-lived objects are likely to live forever. In generational garbage collection, newly created objects are placed under young generation and frequently marked and sweeped. When an object survives, it will be moved to the next generation. Garbage collection runs less frequently on older generations.

4. Python

Python is a dynamically typed programming language. Python is popular due to its simplicity of syntax and the flexibility it provides thanks to dynamically typing. We can use Kivy, a Python framework that helps developers to build mobile and web applications.

There are many implementations of Python. For the sake of argument, we will talk about the most standard

implementation of Python, namely, Cpython. Cpython compiles the python source code into bytecode, then these bytecode are executed with Cpython virtual machine.

Python uses reference counting for garbage collection. Every object in Python stores an information of how many references are pointing to itself. Whenever a reference count reaches 0, the object will be garbage collected. However, this does not garbage collect cyclic references. Thus, Python also implements mark-and-sweep garbage collection algorithm when the memory is low to remove these garbage not collected by reference counting. Notice reference counting algorithm needs protection from race conditions. When multiple threads update the reference count of an object, the count could be incorrectly updated, which leads to memory leak. We could have locks for every Python object, but it is costly to maintain all these locks. Thus, Python uses Global Interpreter Lock (GIL) to lock the entire interpreter whenever any Python bytecode is executed. This entirely removed the concerns of race condition on reference counting, although it also makes Python's multithreading slow because threads will be competing GIL most of the time (i.e. single thread actually runs faster).

5. Dart

Dark is a programming language developed by Google to build mobile, desktop, server, and web applications. Dark has become very popular since Google has released Flutter, an UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop applications. Flutter uses Dart programming language for the development.

Dart is a statically typed programming language, but type annotations are optional because of type inference. Dart is one of those rare languages that supports ahead-of-time (AOT) compilation and just-in-time (JIT) compilation. This is very powerful because developers can use JIT compilation during development and use AOT compilation when they ship the application. JIT compilation enables fast development cycles. With JIT compilation available, we can do hot reload during development. Hot reload allows developers to immediately see the changes they make without compiling the code to native machine code. AOT compilation is used in production because we want the applications to be fast. AOT compilation compiles the code to native machine code (i.e. iOS or Android). Dart can also be compiled into JavaScript, so it can be executed in a web browser. This is beneficial because developers can reuse their codebase for mobile applications to web applications. In fact, the idea of Dart is that developers can develop mobile, web, and desktop applications with just a single codebase. Developers need not to maintain multiple codebases, and thus, changes in one codebase apply to all the platforms.

Dart is sort of single-threaded, but not quite. Dart actually supports multi-threading using what is known as isolates. Isolates are similar to threads but they don't share memory. They only communicate via messages. Note that asynchronous IO code such as future/async/await runs in the same thread and is executed with Dart's event queue (i.e. event loop). However, we should be careful when doing CPU-bound operations with async since async does not unblock CPU-bound operations. In such a case, we should use isolates. Otherwise, the screen will freeze.

Dart uses a variant of generational garbage collection to minimize the overhead of mark-and-sweep algorithm. This is powerful because we don't want to freeze the UI to run the garbage collection.

6. Putting All Together

Recall that we want to make use of Tensorflow Lite on our mobile application to avoid unnecessary network bandwidth. Thus, the programming language we use must support multi-threading. Ocaml and Python are not good at this due to the GILs. However, Python can take advantage of the fact that it is written in C. Python can use tensorflow package, which is written in C, and thus be able to use the multi-threading feature of C. Java obviously supports multi-threading so it's good. Dart supports multi-threading with isolates. Although Dart's multithreading doesn't share the memory, it can communicate to each other with messages. Although sending data this way is slower, the delay is amortized if we consider the time it takes for processing the image with tensorflow (machine learning algorithm is time consuming). Thus, we can get rid of OCaml from our options and think about Python, Java, and Dart.

Python occasionally runs a mark-and-sweep algorithm when the memory is low. If Python runs this in the same thread, this can cause the UI to freeze, which we want to avoid as much as possible because this gives users bad experience and frustration. Even if Python runs garbage collection on different threads, it stills locks the main thread (where UI is running) due to GIL. Java and Dart do not suffer from this issue because they use generational garbage collection. Thus Python is not a good option. Now we are left with Java and Dart.

Speed is a key in our application. Note Java code is compiled to bytecode, and it can be run in JVM. Although this is pretty fast (faster than interpreted language), it is still not as fast as native machine code. Dart, on the other hand, can be compiled into machine code, giving users the best performance of the app. Moreover, Dart can also be interpreted, which makes the development cycle

frustration-free. Every time we change the code, we see the changes immediately using hot reload, which is possible if a language can be interpreted.

Conclusion

Dart is a very good choice for developing the garage-sale buyer app we propose. It supports multithreading so it can execute CPU-bound operations without freezing the UI and keep all the animations. In addition, it uses generational garbage collection, so the overhead of mark-and-sweep algorithm is minimized. Finally, Dart code can be both compiled and interpreted, which provides fast development cycles when we build the app and fast execution of the program in the production.

References

[1] Leler, Wm. Why Flutter Uses Dart. https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf

[2] Tran-Nguyen, TruongSinh. Flutter/Dart Non-blocking Demystify. https://medium.com/flutter-community/flutter-dart-async-concurrency-demystify-1cc739aaae57

[3] The Dart Type System. https://dart.dev/guides/language/sound-dart#type-inference

[4] Daalia, Saurabh. Difference Between Various Implementation of Python. https://www.geeksforgeeks.org/difference-various-implementations-python/

[5] Ajitsaria, Abhinav. What Is The Python Global Interpreter Lock (GIL). https://realpython.com/python-gil/

[6] Driscoll, Mike. Build A Mobile Application With The Kivy Python Framework. https://realpython.com/mobile-app-kivy-python/

[7] Narang, Mehak. Multithreading in Java. https://www.geeksforgeeks.org/multithreading-in-java/

[8] Gamage, A Thilina. Understanding Java Garbage Collection. https://medium.com/platform-engineer/understanding-java-garbage-collection-54fc9230659a

[9] Dharmadasa, Kasun. Garbage Collection: How It's Done. https://medium.com/@kasunpdh/garbage-collection-how-its-done-d48135c7fe77

[10] FAQ - OCaml. https://ocaml.org/learn/faq.html

[11] Multicore OCaml. http://ocamllabs.io/doc/multicore.html

[12] Minsky, Yaron. Madhavapreddy, Anil. Hickey, Jason. Real World OCaml. Understanding The Garbage Collector. https://dev.realworldocaml.org/garbage-collector.html