CS 131 Homework 6 Report

1. Introduction

We are working on an application, GarageGarner, intended to replicate garage sales. The current version is web-based and most of the computation is done in a central server, with only the user interface running on the user's cellphone. The current state of the feature we are working on collects panoramic images and ships them to a central server to process. However, this creates a bottleneck and takes up too much network bandwidth. With the current state of technology, cellphones are much more advanced and have their own machine-learning algorithms.

The new version is supposed to be "less klunky" and distribute the weight of computations between servers and cell phones. The new version should use TensorFlow Lite to run models on cellphones. This report will evaluate Flutter, written in Dart (version 2.7), as a possible solution to the bottleneck.

2. Dart 2.7

Dart is a language developed by Google in 2011. It has C-like syntax, with imperative, object-oriented, and functional paradigms. The language comprises of two options for platforms: Dart Native and Dart Web. Dart Native is used for programs that target devices, such as cellphones, desktops, servers, and so on. Dart Web is used for programs specifically targeting web. For this project, I will focus on Dart Native because it aligns more with the mobile application.

2.1 Dart Native

Dart Native allows for Dart code to be compiled to native ARM or X64 machine code for devices such as cellphones, desktops, or server applications. With Dart Native, the virtual machine contains a just-in-time (JIT) compiler with functionalities such as pure interpretation and runtime optimization. This means that the compilation of the Dart code occurs when the program is executing rather than beforehand and allows for runtime optimization. Another options for Dart Native is the ahead-of-time (AOT) compiler. This is option is meant for when the program is ready for production: ready for publishing to public application stores or deploying to a production backend. AOT compilers allow for code to be compiled before execution to machine code. With the AOT compiler, the program runs in a Dart runtime that has specific memory management systems that handle fast object allocation and garbage collection.

2.2. Garbage Collector

The garbage collector for Dart is generational, maintaining a two-step system to take care of deallocation of memory. This is important for our application because with the use of Flutter, widgets are created are created as they are rendered and deleted when not visible or the state of the application changes.

The first step is called the "young space scavenger" in which objects with short lifespans are handled. This step works by dividing memory in half: an active side and an inactive side. When objects are created, they are allocated memory in the active side. When the active side becomes full, then only the objects that are live are copied into the inactive side. Then the active side becomes inactive while the inactive side becomes active. To distinguish between live and dead objects, the collector starts at the root object, such as stack variables, and recurse through referenceable objects. Any object that cannot be referenced is determined dead. This step in the garbage collector, however, is blocking.

The second step in the garbage collection system is called "mark-sweep" in which objects with longer lifespans are handled. This process works by traversing through the object graph and marking all objects that are still in use. After traversal of the object graph, the garbage collector sweeps through the entire memory and clears memory on objects that were not marked previously. Because most objects in Flutter are short-lived, most of the objects are handled with the young space scavenger, thus this step occurs less frequently. However, some objects are not handled by the first step and must be deallocated by the second step. In this process, the object graph traversal is blocking.

Together, the two steps above are paired with a scheduling algorithm that alerts the garbage collector whenever the app is idle. With the alerts, the garbage collector can run without impacting the performance of the application along with minimizing memory overhead and memory fragmentation.

3. OCaml

OCaml and Dart multiparadigm languages with the functional and object-oriented capabilities. OCaml, howev-

er, is more functional and used more for computations as opposed to Dart, which is used more for user interfaces.

3.1 Advantages

An advantage of using OCaml over Dart is OCaml's type inference capability. OCaml uses an advanced parametric polymorphism and type inference system in which types of a collection can be inferred by types of its elements. Furthermore, with the use of more generic types, functions can be applied to a broader range of collections, independent of the typing of the elements. As for Dart, the language allows for dynamic types through the addition of the phrase "var" before declaring variables, however because variables must have a type before declaring, there is less inference. OCaml uses type checking before compilation thus there is less chance of type mismatch during execution as opposed to Dart that uses runtime type checking which can cause errors if invalid types are not properly checked.

3.2 Disadvantages

Because OCaml is a less popular language in terms of our application, there is less documentation for programmers to use OCaml in a user interface sense. Furthermore, because there is less documentation, it is harder to debug and understand, from personal experience. In addition, Dart has more documentation of concurrent programming. With large libraries that use Futures and Streams, it is much easier to program asynchronous functions in Dart, which is beneficial to our application in reducing the bottleneck.

4. Java

Java is an advanced language with object-oriented language developed by Oracle. The language is statically typed. Java offers generic typing, similar to what Dart offers. Both languages offer garbage collection to handle allocation and deallocation of memory.

4.1. Advantages

Java, unlike Dart, allows for multithreaded applications through the Java Virtual Machine (JVM). Dart, on the other hand, is single threaded. This is an issue because if a function is blocking, then all operations must stop in Dart until the function finishes. Although Dart supports concurrency, Dart programs cannot achieve parallel operations because of the single threaded nature. Beyond multithreading support, the JVM also allows for versatility of deployment by compiling programs into bytecode. Furthermore, Java is a more well-known language, and thus has more programmers providing solutions to issues, more documentation, and more libraries and APIs.

4.2. Disadvantages

For Java, although there is versatility in deployment because of compilation to bytecode, the negative is that the JVM can only be installed on acceptable machines, losing some of the language's versatility. Another issue with a language so popular is that it is slow to update features. An example of this is that lambda functions were not available until Java 8. Because of the slow nature of updates, there may be a longer delay in flexibility, compared to Dart.

5. Python

Python is an interpreted, dynamic typed, object-oriented language. Similar to Java and Dart, this language offers more complex variable creation through objects. Python is one of the most popular programming languages with a wide range of applications, such as image processing to server management.

5.1 Advantages

One of the biggest advantages of Python is the easily understandable syntax. This allows for programmers to quickly learn and develop applications. Furthermore, being a popular language means that there are a larger number of APIs and increased support for debugging. Being an interpreted language means that Python relies heavily on type inference, which can allow programmers to deploy programs faster as less time is spent on minor details of the code. A library that may prove to be an advantage is the asyncio library, allowing concurrent operations within the program. With the asyncio library and extensive documentation, the programmer has much freedom with scheduling.

5.2 Disadvantages

Python has a much more flexible type checking system; however, the drawback is that from reading Python code, it is harder to understand what the type of a variable is without deeper investigation. Dart has a hybrid system that allows generic types along with explicit typing, thus understanding Dart code may be easier. Furthermore, because Python code is interpreted and not compiled, Python loses out on compile optimizations that Dart can provide.

6. Conclusion

Dart, although does not have the multithreading capabilities of Java, is a versatile combination of OCaml, Java, and Python. In terms of our intended application, Dart provides concurrent capabilities, garbage collection, explicit and generic typing, and option between AOT and JIT compilation. Thus, Dart should be used as a possible solution to

mitigate the bottleneck of the current GarageGarner application. Although Dart does not have multithreading capabilities, the ability of asynchronous programming may be enough to satisfy our application.

7. Works Referenced

https://dart.dev/platforms#dart-native-vm-jit-and-aot

https://medium.com/flutter/flutter-dont-fear-the-garbage-collector-d69b3ff1ca30

https://medium.com/dartlang/dart-2-7-a3710ec54e97

https://ocaml.org/learn/description.html

https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html

https://docs.python.org/3/