CS 131 Dart Evaluation (*GarageGarner*)

University of California, Los Angeles

Abstract

Dart is a programming language developed by *Google* for developing apps. In particular, it is designed to allow easy and seamless client-side app development with frontend and backend integration and fast execution via compilation to direct native code. To this end, it has many features built into the language to support building and quickly seeing UI code such as hot-reloading and *Flutter*.

1 Dart and Flutter

Flutter and Dart are an extremely popular toolchain for developing native apps as of now, and there exists several reasons for this. First and arguably foremost is how it makes app development easy. It accomplishes this in several ways.

One of the first things to note is that Dart use a C-style syntax, making it fairly easy for most programmers to pick up. This includes things like being a static and strongly typed language (which can make the language more robust). In addition to that, it is an object-oriented language which make it feel very familiar for the majority of app developers. This is in stark contrast to languages like OCaml, Scheme, or Prolog. General familiarity with the syntax of the language helps make languages like Java, Python, and Dart attractive to developers.

Dart also includes many features that make it easy to use as well. There are a whole slew of examples of this, but one of the most notable ones is the ability to *hot-reload*, or see the effects of modifying code immediately (without having to recompile/restart the app). In additional, it implements a stateful hot reload where the tester's state is maintained even through code changes, making development quick and relativevly painless. In non-interpreted languages (e.g. Java), this capability would be extremely difficult and inefficient to implement (if not impossible). Another example of a popular framework which allows this is *Flask* for Python; when making local changes to the code, the changes are detected and the new behavior is propagated to the webpage. This feature is possible and easy to implement in Dart and Python due to their ability to be run through an interpreter and JIT compiler.

Another large reason why Dart is an excellent choice for developing apps is simply a function of it being made for that purpose. With the addition of Flutter, a developer already has all the tools necessary for app development. To build a simple Hello World app in Dart/Flutter can be done in only a dozen or so lines of code and still look professional. In contrast, using languages like Python or Java require importing many graphics libraries and working much closer to the metal (with Flutter, the SKIA C++ graphics library is already imported and interfaced with, allowing seamless native widget UI programming). Additionally, the choice of using an object-oriented language makes a lot of sense for a UI SDK, as much of the behavior can be modelled simply using object primitives.

One of the last and often overlooked but important features of Dart/Flutter are simply the speed and efficiency of the language and SDK. They achieve high performance several techniques. The first is the ability for Dart to compiled ahead of time to native code. In comparison to Java, which is also ahead-of-time (AOT) compiled, Dart compiles all the way down to native machine code. This makes the language run slightly more efficiently, as there doesn't exist any additional overhead from the Java bytecode JIT compiler/additional runtime checking. Python, OCaml, and Scheme are often run as interpreted code instead of being AOT compiled which require a much more powerful JIT compiler and far, far more overhead. In addition, Flutter implements graphics on top of the SKIA C++ graphics engine, which is pre-compiled to extremely fast and efficient native machine code.

2 Pros and Cons of Dart/Flutter

2.1 Pros

Many of the pros of Dart/Flutter have already been listed above, but I will quickly reiterate them

1. Familiarity/easy to pick up for developers
2. Highly featureful for app development
3. Designed for UI design and app development
4. Very fast

In addition, Flutter is cross-platform compilable to both iOS and Android (and even has web support in beta). These already make it a fairly attractive environment for app development, but there are several cons to note as well.

Another area where Dart shines is in its asynchronous code. Much like Python's new `asyncio` module, Dart supports keywords like `async` and `await`, natively allowing cooperative asynchronous execution. This is incredibly important for an app developer, as starting background tasks and keeping the overall app responsive even as it does expensive computations is a must.

2.2 Cons

One of the first and simplest things to note is that, although it uses a C-style syntax and should feel relatively familiar to most developers, Dart is rarely used outside of app development. This means that for programmers who aren't app developers, it is unlikely they have ever used or encountered Dart and will have to learn the new language.

Additionally, Dart and Flutter are both fairly young. This means that there won't be as much information or support for developers available. In addition, many libraries are either still quite young and untested or even non-existent.

3 Use of Dart/Flutter for *GarageGarner*

Without a doubt, Dart/Flutter would be a very strong choice of framework for any app. The ease of development and support for Android, iOS, and web make it very attractive for building any app to reduce development time and manhours. For *GarageGarner* in particular, the speed and efficiency of Dart and Flutter make it a strong candidate for running *TensorFlow Lite*. It should be noted that the actual library itself is written in Java for Android and Objective-C for iOS, but both these compile down to fairly fast code (Java bytecode and native machine code respectively). Still, removing additional overhead and running the app quickly will help let the app feel responsive and reduce additional computation that could slow down the ML classifications.

In addition, the strong native support for asynchronous code execution would significantly help with running *TensorFlow Lite* in the background. Especially with the use of an AI accelerator, asynchronous execution will allow the device to use its accelerator effectively without bogging down the rest of the device/app.

4 Concluding Thoughts

Dart/Flutter are a strong candidate for *GarageGarner*. The client-focused design for Dart and the Flutter library itself suggest both a quick, smooth development cycle for developers and an effective, simple application for users. Its support for Android, iOS, and web make it extremely flexible and allows developers to only worry about developing one app instead of focusing on and learning native programming for each platform. As a statically and strongly typed language with lots of support from Google, the code should be fairly reliable. And last, Dart's language design allows it to run quickly on every platform, making it efficient and responsive.