Language bindings for TensorFlow

Saman Hashemipour
University of California, Los Angeles

Abstract

Tensorflow, a popular machine learning API is most commonly written in Python. Writing Tensorflow applications in C++ or CUDA code deal with performance bottlenecks due to the high amount of data that is being processed. Since for this project we will be dealing with smaller datasets, we expect the bottlenecks in our code to arise due to the time spent initializing the Tensorflow API code rather than dealing with data. In this report, we will explore 3 alternatives to the more popular Python Tensorflow library. In particular, we will be looking at Java, OCaml, and Kotlin as alternatives languages to use

1. Introduction

Since we have multiple languages and a small data set it would make sense that we look for a language that doesn't take a long time initializing the model that we need to use. When using a large data set the amount of time that we spend initializing the model is rather insignificant but in our case, we spend less time performing computations and the model initialization consumes a large portion of our program time.

In this assignment, we will consider the drawbacks and benefits of 3 other languages compared to Python. Specifically, we will be looking a Java, Ocaml, and Kotlin and their ability to implement Tensorflow models quickly.

It is important to keep in mind why using Tensorflow in different languages can be helpful. Python is the most popular language for using Tensorflow with large datasets, but this is usually because with large datasets the amount of time to initialize the model is negligible. However, for our assignment, we are dealing with smaller sets of data which means a much larger portion of the time for our program will be spent on these initial bindings for Tensorflow. Therefore, to ensure that we are gaining optimal speeds on our dataset we need to take a look at other languages to minimize this overhead.

2. Java

Java is a programing language that was designed to contain minimal implementation dependencies. Java comes equipt with a static type system, as opposed to a dynamic type system in languages such as Python. The all Java code must reside in a class declared in the language. Java code is then compiled into a bytecode that can be run on the JVM (the Java Virtual Machine). There are a few features that Java has that makes it unique and so popular. These features include but are not limited to garbage collection, static typing (mentioned above), automatic memory management, as well as libraries for concurrency and multithreading.

2.1. Advantages

Some of the key features of Java that make it such a good candidate for a implementing an application server herd and Tensorflow is the object-oriented design of Java and the concurrency APIs that Java can use. One of the APIs that gives us the ability to develop an asynchronous event-driven network is

the NIO2 API. This API is almost as suitable as Python's asyncio library for implementing our application server herd. This library works in a very similar way to the asyncio library, by putting messages on an event loop as a callback and dealing with the messages one at a time. This is almost exactly the same as the asyncio library, with the exception of a few name changes for some of the calls that are made. Keeping in mind that Java's NIO2 library is a great option to implement this application server herd we should look at other features of Java that make it such a good candidate for implementing the task at hand.

As we mentioned above (in section 2) Java is executed through the Java Virtual Machine which is what makes the language so portable. This portability is something that could make Java more desirable than Python when implementing this assignment. Since Java has the ability to break things down into bytecode we see that the primary benefit of this is the portability of Java.

Lastly, we know that Java has the ability to run in parallel and gives us the ability to multithread our application, which greatly increases the speed that we can process. Unlike Python, if Java code for this application were written well with the proper parallelization we would have a much more computationally powerful application.

2.2. Disadvantages

We mentioned above that Java lacks dynamic type declarations and must resort to a static alternative. This can make it difficult to implement certain features of the Tensorflow library. This means that the person developing the code needs to put in much more thought and consideration when writing the server side application. Overall, Java is very useful in developing server-side applications, but it comes with trade-offs for performance and simplicity.

3. OCaml

OCaml is a modified implementation of ML. It has many similarities in ML's syntax with minor differences. It does, however, use modern object-oriented structure, which is one of the things that will make it a good candidate for implementing our application server herd and Tensorflow.

3.1. Advantages

One of the main advantages of OCaml is the performance of the language. Somewhat similar to Python, OCaml uses type inference and static typing (more similar to Java). This helps in the implementation of Tensorflow, but it checks the types of the variables and functions at compile time, which helps reduce the number of runtime errors. The obvious benefit for this is that at runtime the types have already been checked and there is little to no need to worry about incorrect type placement throughout the code.

The OCaml compiler can compile directly into an executable but also offers a more interactive top layer that acts much like an interpreter which is similar to languages like Python.

3.2. Disadvantages

OCaml does have a few drawbacks. The type checking method mentioned in the section above does require us to check for the particular type that we are passing to a function which can make the code a little bit more tedious to write. This is one of the things that makes OCaml a language that would be more difficult to develop a working protocol. OCaml also deals with the global interpreter lock, which prevents parallelism, similar to Python. An asynchronous server that creates new threads for every request will have lower throughput than more higher performance languages that support parallelism.

4. Kotlin

Kotlin is a programming language with a similar structure to Java for compilation and execution. Kotlin runs on the JVM (the Java Virtual Machine) and converts its code into portable bytecode which is one of the features that makes it so popular. Kotlin also comes equipt with a number of libraries for asynchronous functions in your program. Kotlin shares a number of similarities to OCaml, Java, and Python along with a few differences.

4.1 Advantages

One of the key features that make Kotlin a good candidate for implementing our application server herd is the portability of Kotlin. Much like Java, Kotlin is turned into bytecode that is very portable and can be run on virtually any machine. Since Kotlin uses the JVM we see that we can use Kotlin to develop our program and maintain a high level of portability.

Kotlin also comes with a library that allows for asynchronous program development. This library, *coroutines*, comes with all the components that we would need for asynchronicity.

Type declaration in Kotlin is also something that makes it a good candidate for the development of our project. We use static declaration with type checking much like in OCaml. This is good because it requires less typecasting and increases the readability and ease of development for our code.

Lastly, like Java, Kotlin has the advantage of using multithreading to improve its performance. This is critical for high load applications and will help us run our code faster with the use of multiple threads. This is one of the things that will lead us to see why Kotlin is one of the best languages of the three we have looked at in this report.

4.2 Disadvantage

Since Kotlin does not have specific bindings for Tensorflow the development of our program may take a little more time, research, and care when in the development stage. Although this is not the biggest difficulty to overcome it does add a layer of understanding that the programmer would have to work through.

5. Conclusion

Looking at these three languages, we see that each have a number of benefits and drawbacks. WIth Ocaml we have type checking which can help with minimizing unwanted forced casting, but it comes with slow garbage collection and a greater learning curve. Java, however, has the drawback of having to use static types, but it is more portable. Lastly, Kotlin has the portability of Java with type checking, making it a more suitable candidate for the development of this assignment. Although, it does come with the downside that it lacks bindings for Tensorflow.

References

[1] "Learn Kotlin." *Kotlin*, www.kotlinlang.org/docs/reference/.

[2] "Global Interpreter Lock." *Wikipedia*, Wikimedia Foundation, 20 July 2018, www.en.wikipedia.org/wiki/Global_interpreter_lock.

[3]Radcliffe, Tom RadcliffeTom. "Python vs. Java: Duck Typing, Parsing on Whitespace and Other Cool Differences." *ActiveState*, 3 Feb. 2019, www.activestate.com/blog/python-vs-java-duck-typing-parsing-whitespace-and-other-cool-differences/.