Language Bindings for TensorFlow

Tian Ye
University of California, Los Angeles

Abstract

TensorFlow applications use a high level language such as Python to set up a dataflow graph before performing the actual computations in a language such as C/C++. Reviewing the server herds in the context of Tensorflow, we want to see if there is a language more suitable than Python for specifying the model that TensorFlow is to compute. We will compare the performance of Java, OCaml, and Kotlin in our investigations of searching for a suitable replacement for Python.

1 Java

Most TensorFlow projects spend the majority of their execution time within the C++ or CUDA code with minimal time spent inside the initialization of the model. However, since we are initializing many small servers, and hence many small TensforFlow models, the overhead associated with the creation of the model becomes significant. In this sense, the language chosen to write the dataflow graph is therefore significant and hence we discuss several replacements for Python. Since a C API seperates the user level code that is written in different languages into the code that is actually executed at runtime, TensorFlow has the ability to support multiple languages while still relying on the same central C API[3]. Therefore we are able to search for replacement languages for Python without worry.

2 Java

Java as a language is designed to have as few implementation dependencies as possible. It feature automatic garbage collection and a static typing system. It is object oriented, meaning nearly all of its code must be written inside the context of an object.

Since Java is statically typed, all variables within the program must be explicitly typed. In the context of TensorFlow, this means that the return types of all functions must be declared, which will potentially slow the productivity of programmers as it forces developers to learn the details of the libraries rather than focusing on only the concepts of TensorFlow such as the dataflow graph. However, this static typing also makes it much less likely that a typing error will slip past the notice of the programmer, meaning that the code will bel more reliable and easier to debug. Furthermore, since variable types are checked at compile time, Java will run faster than a dynamically typed language like Python.

Similarly to Python, Java uses automatic garbage collection to free its memory space; this leads to similar performance to Python in this particular aspect as both approach this particular issue with the idea of accessiblity at the cost of performance.

When it comes to the topic of portability, we must consider that Java is unique in that it lies somewhere between an interpreted and a compiled language. While Java is initially compiled in bytecode, the Java Virtual Machine is able to execute on any computer architecture, even compiling some bytecode sections directly to machine code through the use of the JIT compiler. This allows Java to be extremely portable while still retaining good performace. While the language is still somewhat interpreted and therefore will not be as fast as a language such as C, Java nonetheless provides a solid middle ground between performance and portability.

Java also has excellent support for paralellism, meaning that a Java based server will have more throughput than a single-threaded Python server. However, since Java isn't the most supported language for this particular niche task, implementing asynchronous channels will naturally take more work than a language such as Python.

3 OCaml

OCaml is a modern implementation of the ML family of languages with support for functional, imperative, and object oriented styles, although OCaml strongly pushes for functional style of programming. While OCaml does feature a static typing system, it also automatically infers types. It can be compiled directly into an executable but also supports an

interactive top level similar to an interpreter[4]. While TensorFlow does not explicitly support OCaml, there exists language bindings for OCaml via the TensorFlow C API[5].

Out of the languages we are comparing, OCaml is probably the least flexible due to its strict coding style. Further, the functional programming style is relatively difficult to learn, which may make it more difficult to prototype a working TensorFlow server. Although OCaml's unique type inference can be helpful, at the same time it can also make it difficult to write code that conforms to the specifications of the function; hence, writing a working protocol is probably the hardest to achieve in a language such as OCaml.

OCaml implements a mark and sweep garbage collection method, similar to that of Java. While slower than explicit allocation and freeing, it is also easier for developers to write programs.

When it comes to portability, OCaml is a compiled language, meaning that a new executable will have to be generated for each system architecture the program is intended to run on. However, this also means that the code will be optimized into machine code directly onto the machine. However, functional programming languages are in general more difficult to compile efficient machine code in due to challenges such as the funarg problem. However, due to OCaml's option wrapper, the language as a whole will still be very reliable.

In the scope of parallelism, OCaml is similar to Python in that it is subject to a global interpreter lock, meaning that it cannot achieve true parallelism and can only at best achieve interleaving. Hence an OCaml asynchronous server will not achieve significantly better performance than a Python server due to similar limitations in multithreading in both languages.

Furthermore, the language only supports single core use, meaning while you can attempt to run simultaneous processes, its not well supported. This further detracts from any speed that OCaml may otherwise offer.

4 Kotlin

Kotlin is a multifaceted language that can compile on JVM, JavaScript, or natively. It is similar in coding style to Java; as a matter of fact one of its selling points is the ease of transition from Java to Kotlin. While Kotlin does not have the same general support as Java, its support for coroutines aids the implementation of server herd. Furthermore, its native support for compilation via JavaScript and JVM give it support for TensorFlow as well[2]. This permits TensorFlow to be run on the clientside. Further, since Kotlin can incorporate Java libraries while maintaining JavaScript functionality is a massive boon to the language.

Since Kotlin was designed with the Android platform in mind, Kotlin makes full use of single core processes. This means on Android applications, while Kotlin may be a slightly slower language than Java, it will be able to handle more requests[1].

Further, since Kotlin is a relatively new language it is able to adapt more easily to new technologies than languages such as Java and OCaml. Kotlin already has built in support for lambdas and streams, and its smaller following than languages such as Java means that Kotlin developers do not need to wait for a large user base to catch up to the newer technologies[2].

Finally, the similarity of Kotlin to Java means that errors during development will be minimized due to programmers relative familiarity with the style. A distinct advantage that Kotlin prides itself with in regards to its language and debugging of errors is that it will not give null value exceptions; most variables cannot hold the null value[1].

4.1 Conclusion

While nothing will ultimately beat Python in terms of support for TensorFlow, if we were to consider a suitable alternate, we will probably choose Kotlin. The reasons are for the following: Kotlin is well suited for the proposed idea of running on a given client's Android device. Further, it is object oriented, making it easy for the programmer to adapt to the style as opposed to a language such as OCaml. While OCaml is quite fast in a single core environmnent, it by itself does not have enough utilties to justify using the language whereas both Java and Kotlin have many libraries to access. Further, Kotlin edges out Java in most categories; hence it is the language of our choice for this given task if we were to replace Python.

References

[1] Sagar, Paresh. "Java Vs. Kotlin: Which One Will Be the Best in 2019? - DZone Java." *Dzone.com*, DZone, 18 Jan. 2019, dzone.com/articles/java-vs-kotlin-which-one-will-be-the-best-in-2019.

[2] "Comparison to Java Programming Language." *Kotlin*, Kotlin, kotlinlang.org/docs/reference/comparison-to-java.html.

[3] "TensorFlow Architecture | TensorFlow | TensorFlow." *TensorFlow*, TensorFlow, www.tensorflow.org/guide/extend/architecture.

[4] "FAQ." *OCaml*, OCaml, ocaml.org/learn/faq.html.

[5] Mazare, Laurent. "Deep Learning Experiments in OCaml." *Jane Street Tech Blog*, Jane Street, 20 Sept. 2018, blog.janestreet.com/deep-learning-experiments-in-ocaml/.