Language Bindings for TensorFlow Server Herd

Kanisha Shah, UCLA Computer Science

Abstract

Python is a very popular language because of its flexibility and ease of use. Python is a high-level language with numerous well-maintained libraries to help implement any kind of application. With machine learning algorithms, performance becomes an important factor. Python does not perform as well as expected and for such algorithms and thus there is a need to explore other languages. TensorFlow applications are usually implemented in high level languages like Python based on neural networks or other machine learning algorithms. Usually, the program takes very long to build the model since lot of computation is required. The model is made based on the training data provided. Once the model has been set up, it takes very little time to do the computations required for the results. However, this is not the case with neural networks. Even though these algorithms perform supervised learning, they require extensive computation because of the large number of hidden layers. Since Python does not perform well enough, we will analyze performance and features of Java, Ocaml, and Kotlin with Python for making a server herd application which would rely on TensorFlow.

1. Introduction

There are numerous TensorFlow applications. Majority of them spend most of time performing computations. These computations are performed inside C++ or CUDA code. If there is a large Python application sever herd is being run on a considerable set of servers, there would be very little time spent on initializing the model. However, if the same application were to handle a large number of small Tensor Flow models, the overhead of creating the models would become significant. The time spent running Python code outweighs the benefits of executing the code inside C++ or CUDA. Programming language should also support event driven loops for servers since TensorFlow models would be executed based on the request made by multiple simultaneous clients. Thus, choosing a language is very crucial. Some features that would play an important role are flexibility, sustainability, generality, support for event driven loops, support for Tensor Flow bindings, ease of use and performance. We will now analyze the pros and cons of using four different languages: Python, Java, Ocaml and Kotlin.

2. Program Components

There are two main different components of building this application. Application needs to support event driven server for server herd which can be implemented using asyncio library in Python. Application also needs to support TensorFlow models. These two essential components will be explained more in this section.

2.1. Event Driven Server

The main event loop in each server actively accepts TCP connection from clients and process the commands they receive from them. Each client needs to be served asynchronously. The event driven server explained here is same as the one implemented in Python and asyncio for Project on server herd application. An alternative to this approach could be an application server herd with multithreading. Such an application would need more memory and resources. The implementation for reliable multithreaded server can be considerably difficult.

2.2. Tensor Flow

To be able to support different machine learning algorithms in the application, TensorFlow binding is necessary. To support TensorFlow in multiple languages, an API written in C separates user level code written in different languages. It translates all these other languages into C. Usually, the dataflow graph code is written in Python and C which needs to be translated into C for it to be able to run on hardware. So, translation of the code into C becomes an important part of the program. Small scale applications have greater overhead and translation becomes the bottleneck. So, any language used for the application should setup dataflow graphs for small programs quickly.

3. Python

Python is a high-level interpreted language. Since it is never compiled, it has greater flexibility and is more portable. There is a considerable drawback to it though. It is very slow compared to compiled languages as interpreted byte code instead of machine instructions is executed. Python uses Global Interpreter Lock which is a mutex that protects access of Python objects by multiple threads. It decreases performance and efficiency of the program. Python is usually written in an imperative style, but it also provides support for functional style of coding. Functions such as map and filter are the part of its support for functional languages. Python is easy to use, and the syntax of the language is very straightforward. Python has plenty of well-maintained libraries for all kinds of application. It was one of the first languages to support TensorFlow.

Python has dynamic type checking which means the variable types are checked at run-time and the values are assigned to it during run-time. Python uses duck typing. Arguments are passed from caller to callee but there is no need to explicitly provide the types of the arguments. This type-inferencing saves time and developers only need to focus on what type of values are expected. This increases performance and efficiency. Most of the times, type matches but there is a lot of time wasted on checking the types of variables before diving into execution. Python does not waste time on such type checking. This can be a disadvantage if a programmer relies on types in the declarations and on compiler errors to find out all the discrepancies. Compile-time check can catch more bugs avoiding any errors during run time which could potentially lead to unexpected behavior.

Python implements its garbage collector using reference counts. Each object stored in the memory has an additional field for reference count. Reference count tells us how many times that object has been referenced. When a new object references another object in memory, the reference count of the object in memory increases by one. Similarly, when the object does not reference the object in memory anymore, the reference count of the object in memory decreases by one. This makes garbage collection in Python automatic. Whenever the reference count reaches zero, the object in the memory is freed. Even though garbage collection in Python makes it faster and efficient to free the memory, it has problem with cycles. For instance, A and B reference to different objects in memory. These objects in memory point at each other. Now, if A and B point to something else instead of those objects in memory, the objects will not be freed since those objects point at each other and the reference count is not zero. Python however offers asyncio and aiohttp library for event driven servers and protocols.

4. Java

Java is an object-oriented programming language used for general purpose programming language. Java, unlike Python, is static type checking which means the type of variables are determined at compile-time and it is checked whether the values assigned are of the correct type. Compile-time check can catch more bugs avoiding any errors during run time which could potentially lead to unexpected behavior. This affects productivity of programmers and performance of the program adversely. Developers need to be cautious regarding the types and learn details of the routines they are using to match all the types. Since a check is already done at compile time, it runs very fast during run time.

Java has an automatic garbage collection mechanism set up. Java uses mark and sweep garbage collection. After regular intervals, Java starts at the root block in memory and mark the blocks in the memory that can be reached. After the marking process is finished, sweep process is started. All the unmarked blocks in the memory are then removed. This collection is done in recent generations. The older the generations, older the blocks and thus they are likely to be removed. This is why collection is done in the most recent generation. There is a need to make two passes over the memory to do garbage collection using mark and sweep. Even though Java requires to make two passes over the entire heap, garbage collection is done on a separate thread. This increases performance of Java. Java never leaves unused memory on the heap like Python does because of the cycle problem. Python needs extra memory to store the reference count. Keeping reference count updated at all times needs more operations which decreases performance. Java performs better since garbage collection does not impact the performance of the application.

Java provides support for multithreading which improves performance of an application using concurrency. Support for event driven loops is also provided using libraries such as

Netty and NIO. Multiple threads can be run in parallel to increase efficiency. Java code is compiled on JVM which increases performance and portability. Java is compiled into bytecode. JVM can execute this code on any machine irrespective of its architecture. Java is not as fast as C (compiled language) but is considerably faster than interpreted languages like Python

5. OCaml

OCaml is the functional programming language of the ML family. Despite that, it provides support for imperative and object-oriented style of programming OCaml does not depend on type declarations, it does type inferencing. It has static type checking and can catch any type errors at compile time. OCaml program is compiled into machine code. Since it is directly compiled to an executable, it is run on the machine. This is not good for portability since it is machine specific code. It needs to be compiled separately for distinct computer architectures.

Ocaml has its own garbage collector just like Java. Ocaml's garbage collector performs mark and sweep. At regular intervals, it visits each block in the sweep starting from the root blocks and mark them if they are occupied. It then visits all the blocks again and removes all the non-marked blocks. This garbage collector is very slow and gives poor performance. It has to go through all the blocks twice. Allocation and deallocation need to be done explicitly. Ocaml does not have explicit support for TensorFlow but language bindings do exist for Ocaml. Like Python, Ocaml also uses Global Interpreter lock which does not allow for true parallelism. Implementing event driven loops in a functional language can be difficult. However, there does exist libraries to support such needs. Asynchronous event driven servers can be written in Ocaml using Async library.

6. Kotlin

Kotlin is a lot like Java which is an advantage to developers who already know Java. The learning curve would be smooth for them. Like Java, Kotlin is an object-oriented language. It is a statically typed programming language but allows type inferencing too. Statically typed languages can catch more errors but do not have enough flexibility. Inferencing gives flexibility and prevent any errors. Kotlin has smart casts so that there is no need to do it manually. Kotlin supports multi-threading. It can create multiple threads to run long-running intensive operations. However, Kotlin, unlike Java, has support for coroutines. To complete tasks asynchronously, it can end suspend execution at any point without blocking threads while executing long running intensive operations. It allows us to implement non-blocking and asynchronous servers whose implementation is concise and easy to understand.

Kotlin is based on JVM (Java Virtual Machine). Since Kotlin is compiled into bytecode like Java, it has same the same advantage of portability and performance. This is the reason that both the languages are interchangeable, and they can call code for another language (e.g. Java) in their code (e.g. Kotlin). While implementing in Kotlin, developers can even use the Java libraries for certain functionality.

There are many features that Kotlin provides which sets it apart from Java. Kotlin provides developers the ability to extend an existing class for adding new functionality. Extension can be simply done by prefixing name of class to the name of the new function. Kotlin does not allow us to assign null values to any variable or object which prevents errors of null pointer exception. Kotlin is a mix of procedural and functional programming language. It provides many useful methods for functional programming such as lambda, operator overloading, higher-order overloading and lazy evaluation.

7. Conclusion

Each language has its own strengths and weaknesses. The most suitable language for implementing any application depends on its specification and use. Performance is our primary concern, So, we should choose a language that supports TensorFlow and asynchronous handling of event driven loops. Kotlin can support multiple threads, and asynchronous tasks for server proxy herd. It has TensorFlow support and can use Java libraries for implementation. Considering these features of Kotlin, it should be the clear choice for our application.

8. References

[1] "Event Loop." *https://docs.python.org/3/library/asyncio-eventloop.html*

[2] “Coroutines and Tasks”, *https://docs.python.org/3/library/asyncio-task.html*

[3] “Python Vs Java: Duck Typing, Parsing On Whitespace And Other Cool Differences”, *https://www.activestate.com/blog/python-vs-java-duck-typing-parsing-whitespace-and-other-cool-differences/*

[4] “Global Interpreter Lock”, *https://wiki.python.org/moin/GlobalInterpreterLock*

[5] Java (JVM) Memory Model – Memory Management in Java, *https://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java*

[6] “Java Vs Kotlins”, *https://www.educba.com/java-vs-kotlin/*

[7] “TensorFlow Ocaml”, *https://github.com/LaurentMazare/tensorflow-ocaml/*

[8] “Kotlin Native”, *https://github.com/JetBrains/kotlin-native/tree/master/samples/tensorflow*