Homework 6: Language bindings for TensorFlow

Zixuan Wang, *University of California, Los Angeles*

Abstract

There exist plenty of applications using machine learning algorithms in the real world and thus TensorFlow has become one of the most significant software libraries for developing those applications with machine learning algorithms involved. Considering the importance in high performance numerical computation, our application server proxy herd is based on TensorFlow and we are trying to figure out other programming languages other than Python to avoid wasting too much time to handle small queries. We evaluate the disadvantages as well as the advantages of using Java, OCaml, and Swift to help to make our choice.

1. Introduction

Nowadays, performance is always a critical issue in the area of Computer Science. Choosing appropriate programming language to build up the application is a challenging and important task for software engineering developers. Considering our design of the application server proxy herd, the performance using Python is not up to our expectation due to the large amount of time on setting up models. To solve such problems, we are looking for the effectiveness of three candidates: Java, OCaml, and Swift because TensorFlow functionality is also provided in these programming languages.

We would examine the benefits as well as the drawbacks of writing our program in Java, OCaml, and Swift. Our focus would be on the ease of use, flexibility, generality, performance, and reliability. In the meantime, we are looking for an alternative to support the even-driven servers better because the TensorFlow models in our application server proxy herd would be used to handle the arriving connections and events.

2. TensorFlow

A TensorFlow program is basically written by users in different languages and then is converted to C++, which means TensorFlow is using C++ templates for efficient parallelism operating in CPUs/GPUs. So, TensorFlow is composing either operations or abstractions using the libraries inside after the users have sent their computation graphs.

The prototype we designed for the project needs to perform the connections and process messages in an asynchronous way. We have firstly done this project in Python, applying the asyncio library. However, too many small queries need to be handled to build up or execute the models in the runtime and Python is performing really bad. We want other programming languages that are able to build up the relatively small models compared those in the machine learning in a much faster way.

3. Python

Python is an interpreted high-level language. Python is a famous programming language to use in TensorFlow and we are writing our application in Python initially.

3.1. Advantages

Python makes it convenient to set up our application server herd because it is easy to add servers. Technically, the event loop schedules all the co-routines and each server runs the same code for an event loop. Moreover, the dynamic type checking in Python means that it checks types at runtime instead of at compile time. So, we do not need to know all the types of objects returned when we build up the prototype. Since the various functions involved and the types returned need not to be fully understood, the learning curve for our application written in Python is not steep at all. In fact, Python itself is not a difficult programming language to learn. Besides, TensorFlow has been prioritized Python internally and is originally designed for Python users as well. The nature of Python as a high-level interpreted language represents that our application would be portable for use due to the fact that Python code is not compiled to be an executable file. Python also has excellent readability and the developers can easily understand the written code without much trouble. The automatic memory management is another advantage for Python. The memory management for Python is mainly related to a private heap used to contain all of the data structures and objects. Python is doing the garbage collection using reference counts. During the real implementation, each object would have its own field to keep track of the number of references. The number of references would increase or decrease when the object is referenced and a zero of number of references represents a safe status for future deletion. In this way, Python is able to delete the objects once they are not needed and the instant actions for memory management make Python a good choice for our project.

3.2. Disadvantages

The dynamic type checking may also create a problem for Python developers. Although it is easy to start the project, the difficulty of figuring out the type of objects would be a problem for further steps. Since Python is of a dynamic type checking style and checks types of the objects at runtime, this reduces the reliability of the program and makes it a hard time for the developers to debug. Also, checking types at runtime would cost much more time. So as the memory management. Always keeping track of the number of references is also a great cost and creates an overhead of performance. As an interpreted high-level programming language, Python takes more time to run and the program is relatively slow compared to other non-interpreted languages. What's more, the bad multithread performance of Python makes it a problem to reach the effective parallelism. While Python is using an inefficient way called multithreading Global Interpreter Lock, the mutex used to control access and synchronize threads makes Python less scalable for multithreading servers. Hence, the lack of support for multithreading is also a disadvantage for Python.

4. Java

Java is an object-oriented programming language and has several well-known characteristics such as being object-oriented, based on classes and the concurrency. Java, in addition to being imperative, is used to use objects in the programs.

4.1. Advantages

The Java is a style of static typed, which means that it checks types at the compile time. The static type checking in Java makes it easier for the software engineering developers to debug their programs because we do need to know all the types of objects returned when we build up the prototype. By fully understanding the types of various functions due to the fact that all the types of objects much be declared before we can really use them, it would help us in the future design of our application server proxy herd and the debugging process as well. As a result, this increases the reliability of using Java. Also, checking data types at compile time instead of at runtime would save our time. Java, as a language defined between the compiled language and the interpreted language, is always compiled into bytecode and runs on the Java virtual machine. It is independent of computer architecture and this characteristic makes programs written in Java portable. Java employs a traditional garbage collection method called mark-and-sweep method. Java would start from the initial block for memory to sweep the heap during regular intervals. It would firstly mark the block of memory and remove blocks that are not marked during the second time of sweeping. This increases the accessibility of Java and the automatic garbage collection is useful for our application. As for multithreading, Java is an excellent programming language designed for parallelism. The support for multicore processors in Java makes multithreaded servers possible in our application and raise the throughput. Also, Java has a library named NIO2 API, which is a crucial asynchronous channel API and we can use this library for our application just like using the asyncio library in Python. This API allows us to perform operations asynchronously through the new channels.

4.2. Disadvantages

Java is a static type checking programming language, so the developers need to know and declare all the types of objects before using them in the program. This may become a challenge for the developers to start the project for the first time. Writing in TensorFlow would become much more challenging for developers if they need to know the exact details to build the computational graph. Using NIO2 API may become another challenge for developers because it is more complicated than the asyncio library and also not as high-level as the asyncio library as well. Also, Java's way of garbage collection would decrease its performance and the internal nature of Java would make it not be so fast as other programming languages such as C. This is because Java is always needed to be compiled into bytecode and it should make a compromise between portableness and performance.

5. OCaml

OCaml is a kind of ML-derived language and is a good unification of being functional, being imperative and being object-oriented. The TensorFlow provides some OCaml binding for our use.

5.1. Advantages

OCaml is not only a style of static type checking but also has a characteristic of type-inferring. Both can help the software engineering developers figure out the problems before runtime and improve the performance as well. OCaml is compiled language with good performance but the compiler would also provide a top-level interactive loop, which works like an interpreter. The benefit of OCaml in terms of types is basically the safety of running the program. OCaml's functional features such as the efficient matrix and array operations also make it a good choice for handling clients in TensorFlow and its library that supports asynchronous operations named Lwt makes it possible for us to implement our event-driven servers. Moreover, OCaml is a programming language that puts an emphasis on the performance. Besides the type checking and inferring, the compiler would also perform static program analysis for optimization. This leads to better performance of OCaml compared to other imperative languages.

5.2. Disadvantages

Ocaml is under an ML-like type system and the developers have to be very familiar with the functional programming to write programs in OCaml. The difficulty of writing codes in OCaml is a problem to build our project. Due to the strictness of the type system in OCaml, there would be many constraints to be followed and fully comprehending every exact detail of data types can be a challenge for the software engineering developers. So, it is hard to write our application server herd in OCaml in general. Also, OCaml is not so portable as Java due to the compilation to the machine code. OCaml also cannot achieve true parallelism and cannot support for multithreading servers because it prevents multiple accesses to the same thing as Python and suffers from the global interpreter lock.

6. Swift

Swift is a relatively new programming language and is a general-purpose programming language developed by Apple. The TensorFlow has offer language bindings for our use.

6.1. Advantages

Swift is statically typed checking and has type inference. Besides these, it also employs a read-eval-print loop combined with the feature called playgrounds to enable interactive running and debugging. Its advantages in debugging can be demonstrated well by such characteristics. The developers can be less worried about the bugs occurred in the runtime, thus increasing the reliability. The performance is also superior due to the nature of compiled language and Apple has also done optimizations especially for Swift. The feature of being protocol-oriented is suitable for our application server herd, with applying generic programming. The learning curve is not steep for Swift, considering its close relation with Objective-C. It also allows multithreading, increasing the throughput and making it a good choice for our developers to build the event-driven prototype for multithreading servers.

6.2. Disadvantages

TensorFlow for Swift is a somehow new programming model and it is still under development. Thus, it may exist bugs that cannot be figured out quickly and the codes could not be so much maintainable. Also, as a compiled language, Swift is not so much portable, and Apple designed Swift initially for its own operating systems like iOS. Therefore, despite the fact that it should be operated better for our application focus on mobile users especially iOS users, it may occur other problems for users using Android and other operating systems.

7. Conclusion

As we can see, every language has its own advantages and disadvantages. OCaml and Swift are doing well in terms of performance but OCaml is not easy to comprehend and Swift has a problem with the portability. The trade-offs are always the case in the area of Computer Science and we would like to recommend Swift if our application is heavily focused on Apple mobile users. Otherwise, Java should be a good choice because it is not too bad with respect to performance and can achieve exceptional parallelism Besides the advantages we mentioned above, Java is also a long-established programming language used for building applications.

References

[1] Homework 6. Language bindings for TensorFlow

https://web.cs.ucla.edu/classes/fall18/cs131/hw/hw6.html

[2] TensorFlow. https://www.tensorflow.org

[3] TensorFlow for Java.

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/java

[4] OCaml bindings for TensorFlow.

https://github.com/LaurentMazare/tensorflow-ocaml/

[5] Swift. https://developer.apple.com/swift/

[6] Swift for TensorFlow.

https://github.com/tensorflow/swift

[7] my own report for the project.