HW6 Language Bindings for TensorFlow

CS 131, Programming Languages

Zhouyang Xue

Abstract

The objective of this homework is to research the adaptability and pros/cons of four different languages if utilized to implement a server proxy herd application. The application involves machine learning algorithms and is heavily relied on Tensorflow. This research paper will have a close look at Python, Java, Ocaml, Rust, and how they perform on handling this job. In the following, I would mainly examine five different metrics, which are ease of use, flexibility, generality, performance, and reliability.

1. Introduction

Tensorflow is an open-source machine learning framework. It includes a software library that is designed specifically for numerical computation via data flow graphs. The Tensorflow architecture is constructed on four main laryers, Client, Distributed Master, Working Devices, and Kernel Implementations. Client is a program built by users to initiate graph executions and to communicate with the Master. Master is in charge of all the working devices It's main goal is to partition the graph passed in from Client, look for a specific sub-graph, and distribution sub-graphs among workers. Workers are inter-connected and under the charge of Master (e.g. CPUs and GPUs), and Kernel Implementations are designed for individual graph operations. Therefore, the work flow starts at Client, passed into Master, and distributed among Workers.

2. Python

2.1 Pros

In terms of language adaptability, Python is absolutely the winner to implement TensorFlow related applications (but sadly it has to be substituted). Python was the first client language supported by Tensorflow. So far Python support more features of Tensorflow libraries than any other language. Moreover, Python allows users to import and use C modules, in which the core of Tensorflow is implemented, and all functionalities are available.

Plus, Python is extremely friendly towards users. That being said, Python is very easy to use and read.

2.2 Cons

In this situation, the bottleneck of the application lies under Python module setup. Therefore, to improve the application's overall performance, we need to find a substitute from the following three languages.

3. Java

3.1 Pros

Speaking of Java, its most well-known privileges are its multi-thread programming functionalities and garbage collection. Indeed, Java's advantage in this case is supported by these reputable features.

First of all, in terms of reliability and correctness, Java deserves an A. Java uses a static typing strategy. That is Java check variable types before run-time to eliminate syntax and logical errors. This feature makes the application more reliable and easier to debug.

Furthermore, Java variables are all references, and it utilizes a garbage collection policy that relieves the programming burden from the users. This feature improves its ease to use. Also note that Java uses a garbage collection that deletes unreachable objects, while Python uses reference counter. This difference makes Java a little more efficient than Python from this perspective.

Finally, Java is an object-oriented language, so each graph can be defined as an object, sub-graphs as children of the graphs. In this way, the logic flow of the program becomes very clear compared to other languages. Java's object inheritance also makes it convenient to implement such structure.

3.2 Cons

The disadvantage of this language lies under its flexibility and ease of use.
First of all, to install Tensorflow for Java, the user needs to make sure that Java Virtual Machine (JVM) has been installed before head. JVM is an abstract computing machine that enables a computer to run Java programs. This feature damages the application's flexibility. It means that to develop this application, the programmer has to install JVM, and to use this application, the users also need to install JVM. In addition, to build up models for Tensorflow, Java would most likely need assistance of external libraries to support the models. Therefore, it becomes the programmer's responsibility to make these extended libraries available in JVM, and also to make that these libraries are compatible with others. This feature increases a Java Tensorflow application's complexity, and damages its flexibility furthermore.

In addition, Java does not use duck typing. While it improves the application's reliability, it also makes the program less easy to write.

4. OCaml

4.1 Pros

OCaml is a good choice to implement a reliable application. Similar to Java, OCaml uses static typing, and detects syntax error before runtime, which eliminates potential logic errors and makes debugging easier.

Moreover, in terms of generality, OCaml is machine independent so that it is compatible with various platforms. This feature also allowed OCaml to be ported to multiple platforms concurrently to achieve parallelism. This improves the application's performance and stability when it comes to dealing with large amount of data.

Finally, in terms of popularity, OCaml is also a good choice. As a functional programming language, it is very similar and related to Lisp. Lisp has a long history in application in the machine learning field. Therefore, many machine learning programmers are fairly familiar with the syntax and logic when getting into OCaml.

4.2 Cons

Two major issues of OCaml are ease of use and performance. First of all, OCaml is a functional programming language that cannot avoid the fact that it is really difficult for TensorFlow to support as a Client language. As addressed before, the core of TensorFlow is implemented in C, thus to use the core API of TensorFlow, one needs to integrate C module with OCaml. In this case, if I choose OCaml to replace Python, I would use the OCaml-Ctype for C binding. Therefore, the

complexity of the application significantly increases, and makes the whole program really tough to implement.

As for performance, OCaml has poor floating point performance from x86 code gen, and has a 16MB limit on 32-bit systems. It is neither designed for to support multicore operations, which makes it less competitive in on newest computers and for machine learning tasks that normally involves multicore processing.

5. Rust

5.1 Pros

Rust is a rapidly evolving programming language that has the performance and control of a low-level language, and a powerful abstraction of a high-level language. Rust is a winner on both reliability and performance. Rust guarantees memory safety without garbage collection. And this is by far Rust's most significant achievement. For most languages, the approach towards memory safety is by applying a GC, such as Python and Java. However, using GC inevitably compromises the runtime performance of the application. Instead of garbage collector, Rust uses a borrow and ownership system. This feature improves Rust's reliability without damaging the application's runtime performance. Moreover, Rust supports concurrency without data races, which is another example of boosting efficiency while maintaining reliability.

Rust also supports composition over inheritance, which I found very useful for this application. If graphs are defined as classes/objects, this feature makes it very easy to divide the graph into sub-graphs with inheritance, and to merge sub-graphs via type composition.

5.2 Cons

Popularity (Generality) is a big issue for Rust. Most programmers are not familiar with this language, which implicitly makes Rust code difficult to understand. This would also lead to challenges in co-programming on the project. Also, since Rust programming language is evolving in a very fast pace, the older version might not be compatible on newer platforms, which could be potential problem.

6. Conclusion

My recommendation among these three programming languages to replace Python is Rust. First of all, Rust has better performance and reliability by absorbing advantages from both Java and OCaml. It's also very adaptable to this particular application, which heavily involves data processing and TensorFlow. As for the disfavor on its popularity, my point is that it should be considered trivial compared with Rust's privileges.

7. Reference

[1] Installing Tensorflow for Java
https://www.tensorflow.org/install/install_java

[2] Pros and Cons of OCaml:
http://flyingfrogblog.blogspot.com/2011/03/pros-and-cons-of-ocaml.html

[3] The Rust Programming Language
https://doc.rust-lang.org/book/second-edition/ch01-00-introduction.html

[4] Rust, an Anti-Sloppy Programming Language
http://arthurtw.github.io/2014/12/21/rust-anti-sloppy-programming-language.html

[5] Links given on the spec