Evaluation of Language Alternatives for Using TensorFlow

Computer Science 131, Homework 6

Michael Wu
UID: 404751542

June 6th, 2018

1 Introduction

TensorFlow is a technology developed by Google that aids in distributing computations across different machines. It is intended for the purpose of machine learning, and we wish to use it on our server herd that I prototyped earlier. TensorFlow models computations as graphs, which are submitted by a client to a master process. The master process then splits up the graph to be performed on multiple workers. After the workers are done, the workers are able to communicate with each other to share information and complete the computation [1]. Because it appears that the Python TensorFlow API is slowing down our code, we wish to evaluate alternative language options for our server. We will consider using Java, OCaml, and Vala.

Note that using an alternative language may limit our capabilities in TensorFlow, because "support for gradients, functions and control flow operations ('if' and 'while') is not available in languages other than Python" [2]. Additionally, APIs in other languages may not be forward or backward compatible, and may not follow stability guarantees [3]. This is a serious consideration if we wish to maintain the reliability of our application, as we may require significant additional development efforts when an update to TensorFlow comes out.

2 Java

The Java API works by using a `Graph` object that can be passed around. There also is a `OperationBuilder` class, which seems to use a factory pattern to create an `Operation` that can be executed [4]. The Java API uses the Java Native Interface (JNI) framework to execute some functions written in C++. This API offers a nice object-oriented approach to designing our application, as Java was created for this type of programming.

If we were to reimplement our entire server architecture on this platform, we would also be able to do event-loop driven programming using a library such as Netty [5]. Because Java is compiled into bytecode, it will most likely outperform an interpreted language such as Python. However, both are still relatively high level and would be outperformed by languages such as C++. Thankfully the JNI allows the Java API to use some C++ functions, which would increase the library's performance.

3 OCaml

A TensorFlow API for OCaml exists and provides modules to build and execute graphs. It seems the main module to run a job is the `Session` module, and operations can be added with the `Ops` module. There is support to easily create neural networks and machine learning models [6]. Example code shows the API following a factory design method as well, using the `|>` OCaml syntax to apply functions successively. OCaml can be compiled and would be pretty fast, but as a lesser used language it would most likely suffer from a lack of support. Since many more people use mainstream procedural languages like Python and Java, there are probably better libraries in those languages than in OCaml. So developing our application in OCaml may mean that we will have to write a lot of code from scratch that we would not have to in a mainstream language.

To implement our server in OCaml, we would have to use the Async library to implement an event loop [7]. This introduces a `Deferred` module that indicates a value that will be computed sometime in the future. This is roughly analogous to the `Future` type in Python. Introducing this type of library would mean deviating from the functional programming paradigm, where variables have no state associated with them. The `Deferred` module inherently includes states, because it represents some operation that can either be complete or blocked. So using OCaml is possible, but it is inelegant.

4 Vala

Vala is a high level object-oriented language that is used to generate C code, allowing for fast coding while maintaining performance. It includes support for interfaces, lambda expressions, type inference, generics, and exception handling [8]. It is mainly used to write graphical applications for GNOME, a Linux GUI environment. Vala can integrate easily with existing C libraries, which makes it a good candidate for our server application. Since C is one of the most widely used general purpose programming language, this is a major plus.

Using TensorFlow in Vala requires the use of a library that is described as "highly experimental" and maintained by a single contributor [9]. It seems that these bindings do not receive a lot of support, so it may be unwise to rely on them in production code. They mostly serve as a wrapper around the C API, so we could just write our code in C to overcome the performance problems we are experiencing with Python.

Using Vala could work, but we would most likely have to do some programming in C to implement everything we would like on our server. C should be able to support an event loop driver server.

5 Conclusion

Overall, I would recommend to try out Vala because it could potentially have the largest performance benefit. Although it may require working in C in order to take advantage of existing library support, that should not be too much of an issue. C is a widely used language which is well known for its speed and reliability, and we would be able to take advantage of that by using Vala. The TensorFlow bindings for Vala seem to be relatively simple, as they interface into the C API. Although there is only one contributer to the Vala binding repository I found, we could maintain the Vala TensorFlow repository ourselves if we need to add support for any new features.

If developing in Vala is too troublesome to learn, Java would be a good alternative. The Java API is maintained by Google employees, and contains helpful example code. Java would most likely perform better than our current Python implementation of our servers. It is also well suited to the TensorFlow model, as it was designed for object-oriented programming.

I would not recommend OCaml for our servers, as the functional programming paradigm does not work well when writing a server. Functional programming is good for calculating things and mathematical applications, but the concept of state is not easy to deal with in a functional language. The TensorFlow model deals with passing around graphs and different workers communicating with each other, which does not synergize naturally with OCaml.

We should use either Vala or Java to overcome the performance bottleneck that the Python TensorFlow API causes on our servers.

References

[1] TensorFlow Architecture Documentation. `www.tensorflow.org/extend/architecture`.

[2] TensorFlow in Other Languages. `www.tensorflow.org/extend/language_bindings`.

[3] TensorFlow Version Compatibility. www.tensorflow.org/programmers_guide/version_compat.

[4] TensorFlow Java Overview. www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary.

[5] Netty 4.x User Guide. netty.io/wiki/user-guide-for-4.x.html.

[6] OCaml Bindings for TensorFlow. github.com/LaurentMazare/tensorflow-ocaml.

[7] Concurrent Programming with Async. realworldocaml.org/v1/en/html/concurrent-programming-with-async.html.

[8] About Vala. wiki.gnome.org/Projects/Vala/About.

[9] TensorFlow for Vala. github.com/arrufat/tensorflow-vala.