Language Bindings for TensorFlow

Millie Savalia, UCLA COM SCI 131 Winter 2018

Abstract

Following up on the project, we now want to deploy the the application server proxy herd on a large set of virtual machines. The application uses machine learning and relies heavily on TensorFlow. The prototype uses Python and runs well on large queries but the application seems to be spending most of its time executing Python code to set up the models. This paper examines the feasibility and efficiency of using another language instead of Python such as Java, OCaml, and Rust.

1. Introduction

TensorFlow is a leading open source software framework that helps you build and train neural networks. It is a software library developed by Google Brain Team within Google's Machine Learning Intelligence research organization, for the purposes of conducting machine learning and deep neural network research. It combines the computational algebra of compilation optimization techniques, making easy the calculation of many mathematical expressions where the efficiency issue is the time required to perform the computation.

TensorFlow was designed for large-scale distributed training and inference but is also flexible enough to support experimentation with new machine learning models and system-level optimizations. Although TensorFlow has a slightly messy interface and can be difficult to use at times or slow, it is generally more scalable than many other frameworks and very compatible with distributed execution. It also supports single GPUs to massive systems that involve heavy distributed reinforcement learning with real-time trials and errors. TensorFlow works well with models involving up to hundreds of distributed computing nodes. A lot of people prefer it because it also uses the same code for running on a single GPU or CPU, proving witness to its scalability.

2. Language Bindings

2.1 Java

One of the most popular languages in many different applications, Java is an old age language with its strengths in the object-oriented paradigm of working. It allows developers to run a compiled Java code runnable on all platforms with Java support without the need for recompilation. Java code is compiled to bytecode that is runnable on Java Virtual Machine (JVM) on any computer architecture.

One of the advantages of Java is that it has static type checking, which can prove to be very useful in writing code during the development process. It makes it easier to catch type-related errors during compile-time rather than runtime. The JVM also enables applications to work cross-platform by abstracting OS-specific system calls away from the application code. You can use Java in TensorFlow to make predictions similar to how it is done in Python. Values are passed in for all the placeholders and the output node is evaluated. However, in Java you have to know the actual names of placeholders and output nodes. TensorFlow in Python calls on C++ implementation to carry out actual work. With Java, we can use Java Native Interface (JNI) to directly invoke C++ and use that to create graphs and restore weights and biases from the Java model.

The TensorFlow website acknowledges that the TensorFlow Java API is not covered by the TensorFlow API stability guarantees. TensorFlow does not support "inline" matrix operations, but forces you to copy a matrix in order to perform an operation on it, which can be costly when copying large matrices. Using Java for this application doesn't seem to take advantage of hardware acceleration and distribution. It would not be optimized for speed and resource management like GPU.

2.2 OCaml

OCaml is primarily a functional language with support for object-oriented design along with imperative-style code. OCaml's strengths lie in its conciseness and speed of its code.

Similar to Java, OCaml uses strong static typing with type inference. This can be a major advantage for catching type-related errors at compile time, allowing for more reliable code without requiring type notation as verbose and prevalent as Java. OCaml is also compiled down to bytecode that could be made specific to the system or to an interpreter in the OCaml toolchain. This makes its code highly portable and machine agnostic, which would be beneficial in using it with TensorFlow. OCaml also comes with some very powerful libraries that allow it to interface with low-level details such as the unix library or sys library. The code for most operators is automatically generated from the TensorFlow protobuf specs. Some operators are not supported and the API is not fully developed, but it can already be used to train a convolutional neural network to recognize digits with great accuracy.

A barrier to using OCaml is getting accustomed to the functional paradigm of thinking. One would have to get used to this functional nature in order to fully utilize its capabilities. Additionally, OCaml is not a very popular language as compared to the other languages being discussed here. This would make it all the more difficult for developers to use it for TensorFlow because its low popularity would make it difficult for them to find skilled OCaml developers and actively maintained libraries. Another disadvantage is that LXC does not provide native API for OCaml, which could cause security and reliability issues when using third-party APIs. Communication between a program written in OCaml and low level applications is also difficult since OCaml is not designed as a low level language.

2.3 Rust

Rust is a low-level language best suited for systems, embedded, and other performance critical code. It is a statically typed compiled language meant with its main purpose being to promote memory safety. Rust code will not compile if it has dangling pointers, buffer overflows, or other memory errors.

A major advantage of Rust is that with no runtime overhead, the final code is as fast as C or C++ but much safer. Rust runs fast, prevents segfaults, and guarantees thread safety as a systems programming language. Cross-compiling is also made easier for a Rust executable with a working toolchain and built TensorFlow library. Rust also has a robust foreign function interface.

The Rust API for TensorFlow is also unstable which could cause problems. It is also especially unstable because the underlying TensorFlow C API has not yet been stabilized as well. Since Rust is a relatively young language, some believe it slightly harder to learn and believe it to have fewer resources than a more well developed language.

3. Conclusion

After researching the three languages as alternatives for Python with TensorFlow, I think I would go with Rust. Although Java and OCaml code gets compiled to bytecode making

portability easier, they would incur heavier costs in application itself. Rust seems to be a good fit for developers to pick up due to its suitability for performance critical code, memory safety, and robust foreign function interface.

4. References

Lakshmanan, L. (2016, July 19). How to invoke a trained TensorFlow model from Java programs. Retrieved March 16, 2018, from https://medium.com/google-cloud/how-to-invoke-a-trained-tensorflow-model-from-java-programs-27ed5f4f502d

Mazare, L. (2018, February 18). LaurentMazare/tensorflow-ocaml. Retrieved March 16, 2018, from https://github.com/LaurentMazare/tensorflow-ocaml/

{{metadataController.pageTitle}}. (n.d.). Retrieved March 16, 2018, from https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781786468574/1/ch01lvl1sec9/tensorflow--a-general-overview

On, J. (n.d.). Josh On Design. Retrieved March 17, 2018, from https://joshondesign.com/2014/09/17/rustlang

TensorFlow Architecture | TensorFlow. (n.d.). Retrieved March 17, 2018, from https://www.tensorflow.org/extend/architecture

Tensorflow/rust. (2018, March 03). Retrieved March 17, 2018, from https://github.com/tensorflow/rust