CS 131 Homework 6: Language bindings for Tensorflow

David Chen, 904622289, *UCLA*

Abstract

Tensorflow is a software library developed by Google primarily for the purpose of machine learning and deep neural networks, though it has other uses as well. When running it, most common bottlenecks experiences occur in the C++ or CUDA code, due to the large data volume, large amount of calculations, and GPU intensive tasks. However, in our application, which instead handles many small queries, the main bottleneck occurs in the Python frontend while setting up models. While Python is the most widely supported frontend for Tensorflow, other front ends are also supported, either by Google or third-party developers. library. In this paper, we will examine the Tensorflow library in relation to our application and alternative frontends that may help with our current bottleneck, namely Java, OCaml, and Julia.

Introduction

Tensorflow is built in C++ and CUDA using the Eigen numerical library for high efficiency numerical processing and cuDNN, NVidia deep neural network primitives. [1] It is mainly inspired from natural computers such as the brain and their ability to process large amounts of data and learn what the data means. It has a variety of applications often associated with pattern recognition such as voice analysis, image recognition and processing, or other extrapolations of meaningful information from large data pool. These are tasks seemingly simple for a human to do while incredibly intensive for a computer.

1. Python Overview

While Python is the current bottleneck in our code, it was not chosen as one of the main supported languages for Tensorflow without reason. Python is one of the simplest languages in terms of syntax and the development time needed to get code up and running is relatively short in comparison to other languages. Tensorflow is used for things such as deep learning, so it's not guaranteed that the user will be deeply involved with computer science; many people using it or who have picked it up may have been originally data scientists, familiar with Python and Python-like languages. Python has also long been a popular choice for machine learning, having other machine learning libraries like scikit, so this may have driven Google's prioritization of the Python interface.

In addition, despite being the bottleneck for our example program, this is usually not the case, as the internal processes of Tensorflow take much longer, so Python's slower performance will usually not be significant. Rather, Python's large amount of data science libraries allow for preprocessing and easier handling of outputted data; numpy, scipy, pandas, these are just some of the more commonly used ones for data sciences.

However, because our application deals with large amounts of smaller inputs, the creation of models before passing those models to the C++ core is slowing down our overall execution. It should be noted that many functionalities are being moved into the C++ core and exposed by a API so that other languages can be more easily supported; with that in mind, we will explore some other options for Tensorflow. [2]

2. Java

2.1. Overview

Java is a well-established language primarily based around the concept of object-oriented programming; this means it follows the paradigms of object orientation, namely abstraction, encapsulation, polymorphism, and inheritance. It is class-based, runs concurrently, and has static typing.

2.2. Analysis

Starting from development point of view, Java's static typing will bring the usual set of advantages and disadvantages. While it allows for bugs to be caught earlier on with type-checking, it potentially hinders development and makes code more complex. However, letting bugs be caught earlier on usually makes then easier to fix than having to trawl through code which runs, yet exhibits undefined behavior.

Java's compilation method also has many advantages; Java compiles down to byte code with a "Write Once, Run Anywhere" (WORA) guarantee, meaning that once compiled, it should be able to run without having to work about such thing as hardware. This is because of the Java Virtual Machine (JVM), which is a portable platform which enables the WORA guarantee. This does mean however, the machine the code runs on must have JVM. This becomes more of an issue the more distributed an application may have to be.

Concerning the main worry, of performance, Java does have reputation of being slower and requiring more memory than other compiled languages like C is still has the advantage over Python. Neither are particularly suited for high performance, but Java's "Just in Time" compilation allows Java to compile the byte code as the program executes allowing it much more efficiency. In addition, Java employs concurrency support, which allows us to take advantage of multi-core processors to improve performance. Python in comparison is more sequential by nature. Switching to Java will most likely help with the bottleneck in model creation. [3]

There is one key weakness of using other languages outside Python for Tensorflow, however, and that is the level of support and functionality it has. New language bindings should be built on top of Tensorflow's exposed C API, but many functionalities are still only available in Python and not C. For example, if one were to use Java, one would still have to train the models using Python, as Java can only run predefined ones. [2]

3. OCaml

3.1. Overview

OCaml is a strongly-typed functional programming language which also supports the object system. It also supports type inference, polymorphism, tail recursion and pattern matching.

3.2. Analysis

Depending on the programmer, OCaml programs can be simple to write or mentally taxing. On the plus side, type inference saves time guaranteeing types and one must only check if the code performs operations on the correct data. All type errors with be caught at compile time, which make for strong data integrity and safety during execution. However, if one has somewhat more complex OCaml code, without familiarity with the language one will find the error messages potentially unhelpful and confusing, as they are one, translated from French and two, often print out a string of expected types which are extremely difficult to read. In addition, functional programming is counter intuitive to many programmers who have gotten used to imperative languages, though people who are used to mathematical models may be able to adapt much more quickly.

One strong suit of OCaml in particular, especially compared to other meta languages. Functional programming languages are usually characterized by their comparatively reduced efficiency, especially compared to C in terms of CPU and memory management. OCaml in particular claims to have speed less than, but comparable to C when it comes to intensive numerical computations. [4] This may come to be an advantage if there is a non-negligible amount of preprocessing.

Finally, there is an issue with maturity and popularity of OCaml. The community is of a fewer number and while it does have a package manager (OPAM) and some powerful libraries, it is not as "Batteries included" as say, Python or Java. In additional the OCaml bindings for Tensor flow is not particularly polish and has many known errors.

While OCaml code can be written in a concise and clean manner, it takes experience with the language to do so, so may not be a good choice for reimplementation of existing code if one is not familiar with the idea of functional programming.

4. Julia

4.1. Overview

Julia is a high-performance language focused on numerical and scientific computing. It boasts a large number of mathematical libraries and integrates C and Fortran libraries for algebra, signal processing and string processing.

4.2. Analysis

Immediately one of the advantage for Julia as an alternative would be its similarity to Python in terms of (math -friendly) syntax, duck typing, and automatic memory management, though it does have some key differences. Some of which include it being a JIT compiled language and being one-index. Notably, Julia's

Tensorflow bindings are made to be similar to Python's. [5]

Julia attempts to incorporate the best parts languages like C & Python while maintaining a focus on performance. It can interface directly with external libraries from C and Fortran and directly with Python code. Thus, it maintains many of the benefits of using Python while boasting a default speed exceeding Python while approaching C's when it comes to data processing and being faster than both in handling inputs and outputs. However, Python can be optimized to run faster than Julia using PyPy or CPython, but Julia will be faster without the use of any external libraries significantly improving development time. [6]

Unfortunately, Julia is not without its weakness, largely due to its youth, currently being only at its 0.6.2 release. Due to this, it lacks a sufficiently large community and also is reliant on existing Python and C libraries. Reliability is also an issue as is startup time, though this is less of a factor in running our application, but rather in its upkeep.

5. Conclusion

Out of the mentioned language none seem to be a suitable replacement in its entirely for the purpose of this project. Thus, I believe it may be best to integrate another language into our existing data flow to take advantage of Python's existing infrastructure and having another language handle the model creation. Barring this, for fear of excess complexity, the next best alternative may be Julia, due to its efficiency and computational-power driven development. Unfortunately, it is very young and prone to reliability issues, making it unsuited for production-level applications. It can make up for this to an extent by making use of the C and Python libraries, but even with mature libraries, the core language may not be the best choice. Thus, Java my ne the best choice, though it must be noted that Python will still have to be used to training, while Java will run using the already trained models.

6. References

[1] “Tensorflow Architecture” https://www.tensorflow.org/extend/architecture

[2] “Tensorflow in other Languages” https://www.tensorflow.org/extend/language_bindings

[3] “Java vs. Python” https://blog.appdynamics.com/engineering/java-vs-python-which-one-is-best-for-you/

[4] ”Which programming languages are fastest” http://benchmarksgame.alioth.debian.org/u32/which-programs-are-fastest.html

[5] “Julia vs. Python: Julia language rises for data science” https://www.infoworld.com/article/3241107/python/julia-vs-python-julia-language-rises-for-data-science.html

[6] “Julia Micro-Benchmarks” https://julialang.org/benchmarks/