CS 131 Language Bindings for TensorFlow

University of California, Los Angeles

1 Abstract

Machine learning applications can require an enormous amount of expensive computations when training models. TensorFlow is one particular library that allows us to perform such computations. Typically, the bottlenecks of TensorFlow applications occur inside the C++ or CUDA code, but in our application we have discovered the bottleneck is in the Python code. This paper explores alternative languages in order to speed up the performance of our application.

2 Introduction

Our application running on a server proxy herd relies heavily on TensorFlow. Because our application is fairly simple and small, the overhead of setting up the models with our Python code is the bottleneck of our application. With a larger application, the overhead of initializing the model is far less significant in comparison to the other computations that the application has to do. Therefore, while TensorFlow for Python may be an excellent choice for larger applications, it may not be the best choice for our purposes.

Because the current prototype is being bottlenecked by the time spent executing Python code to set up the models, we are looking to speed up performance by converting the Python code into another language. In particular, we are going to look into the possibility of using one of three other languages: Java, OCaml, and Kotlin. We will consider their advantages and disadvantages as compared to Python in order to see if using one of these languages could reduce the time setting up the models.

3 TensorFlow for Python

TensorFlow is a widely used, open-source library used for research and production, offering machine learning functionalities to users for desktop, mobile, web, and cloud. In Python, it has several modules within the library to support a wide variety of machine learning tasks.[1] Python was the first client language that TensorFlow supported, and currently supports most of TensorFlow's features.[2] For these reasons, it is unsurprising that many who opt to use TensorFlow do so using the Python library.

3.1 Benefits

TensorFlow uses C++ and CUDA to do much of the expensive computations required for the machine learning algorithms that it implements. However, the client-side API is available in Python, which is a much more friendly language to programmers. The API written for Python clients is intuitive and easy to use. Python also has many libraries to support plotting results and manipulating data, such as Numpy and Pandas. Using these libraries in conjunction with TensorFlow modules allows programmers to conveniently run machine learning algorithms and visualize the results in one single Python script.

In addition to specific TensorFlow support, Python itself has many advantages as a language. One such advantage is its dynamic type checking; this makes it easier for programmers to use TensorFlow objects without having to figure out exactly what they are and what all the rules are for these objects. Python simply allows programmers to use them and pass them to other functions without specifying exact types, so the programmers don't have to worry about these details while implementing their application. Additionally, Python's memory manager uses reference counts, sweeping up most garbage immediately when it is available. Overall, Python is a very intuitive, straightforward language with many libraries and capabilities that makes it easier to use on the programmer's end.

3.2 Drawbacks

While Python has the most support for TensorFlow and a great number of features that make it simpler to use for the programmer, these benefits come at the cost of performance. For example, dynamic type-checking in Python means it has to do these checks during runtime, causing it to perform significantly slower than a programming languages that performs type-checking during compile time.

Python also has to keep track of reference counts for its garbage collection. This means that whenever an object is assigned, Python has to do some work to increment a reference count number and also use extra space to store these reference counts. This extra time and space is an extra cost for Python that many other

programming languages do not have.

4 Java

Java is another popular language used by many applications. Java code is compiled into Java bytecode, which can then be translated into different kinds of machine code. TensorFlow provides a Java API, and we will discuss this API and compare its benefits and drawbacks as compared to TensorFlow's Python API.

4.1 TensorFlow for Java

According to the TensorFlow website, the Java API that TensorFlow provides is useful for loading models that were created with Python and then running them in a Java application.[3] The API, however, is currently experimental, and TensorFlow publicly announces on its website that it is not covered by TensorFlow API stability guarantees.[4]

4.2 Benefits

Java as a programming language has many benefits over Python. One benefit of Java is that it does compile-time type checking. While this makes it more difficult on the programmer's end to write a working program, this means fewer runtime errors related to type errors, and it also means better performance because the program doesn't have to check types while it runs. This means that an application written in Java is typically faster and more reliable in this regard than an application written in Python.

Similar to Python, Java has a built in garbage collector. However, Java does not use reference counts. It instead uses a mark and sweep method for garbage collection, meaning that the programmer does not have to worry about allocating and deallocating memory themselves.

Additionally, Java has the advantage in that it is a highly portable language. The Java program is compiled into bytecode, which the Java Virtual Machine can translate into machine code for any computer architecture using the Just-In-Tie compiler. While having to compile while running can hurt performance, the performance is still quite good in addition to the bytecode being extremely portable.

Finally, Java also has support for multithreading. Python does support multithreading to some extent, but this multithreading is bottlenecked by the Global Interpreter Lock, which prevents multiple threads from executing the same Python bytecodes at the same time. Java's multithreading capabilities may make it more suitable for an application that needs to support event-driven servers.[5]

4.3 Drawbacks

Many of the performance benefits that Java offers when compared to Python comes at the expense of more effort on the programmer's side to write the code for the application. For example, Java's static type checking requires the programmer to know and to some extent understand the structure of TensorFlow objects and how they are used.

5 OCaml

According to the OCaml website, OCaml can be used as a general-purpose programming language that emphasizes safety and expressiveness.[5] It is a programming language that is both relatively simple and powerful, with a unique type system to help catch type errors.[5] In this section, we will discuss the benefits of drawbacks of OCaml over Python, both as a language in general and also with regards to TensorFlow support in the language.

5.1 TensorFlow for OCaml

There is a tensorflow-ocaml project available online to provide OCaml bindings for TensorFlow. According to its documentation, not all of the functionalities of TensorFlow are available for OCaml and there are some bugs in the code.[6] However, the project can still generally be used as a means for developers to use TensorFlow with OCaml.

5.2 Benefits

OCaml has many advantages to Python in terms of the way that it works as a programming language. One such advantage it has over Python is that it is statically typed, meaning that all type checks happen at compile-time. OCaml's type-checking system is also advantageous over Java because the types are automatically inferred by the compiler, and the programmer using OCaml does not have to explicitly state the type of each variable. This makes programming easier on the developer's side while also maintaining the performance benefit of static type-checking.

OCaml gets compiled into machine code before it gets executed, allowing it to run very quickly on the

machine as it does not need to be compiled any further during runtime. This leads to better performance during runtime in this regard when compared to Python and Java, although it means OCaml compiled code is not portable like Java is.

Like Python and Java, OCaml also has an automatic garbage collector. OCaml uses a generational garbage collector as well as a mark and sweep method, as opposed to Python's reference counts.

5.3 Drawbacks

One of the most important drawbacks of using OCaml with TensorFlow in our application is that TensorFlow is more easily used in imperative languages such as Python, and it may be more difficult to support it in a functional language such as OCaml. The tensorflow-ocaml project even specifically states that using this library may lead a developer to encountering segfaults, that not all TensorFlow functionalities are supported, and that the API is likely to changed in the future.[7]

Additionally, OCaml, like Python, supports multi-threading but is bottlenecked by its global interpreter lock, which prevents multiple threads from running the same OCaml code at the same time.

6 Kotlin

According to Julius Kunze's blog, the programming language Kotlin compiles to the JVM. When writing my own Kotlin code for the everyNth function, I noticed that when I compiled the Kotlin it compiled into a jar file that could be run either with Kotlin or Java.[9]

6.1 TensorFlow for Kotlin

The TensorFlow backend can be used with Kotlin by following instructions form TensorFlow documentation on building a TensorFlow client in a new language.[9] Following the instructions in Julius Kunze's blog[9] for setting up the TensorFlow client for Kotlin is fairly easy to understand and straightforward.

6.2 Benefits

Kotlin has many advantages over Java. Some of that Java had that Kotlin addressed are that Kotlin has no raw types nor checked exceptions.[10] Kotlin has many of the same advantages over Python that Java has; it is compiled into Java bytecode that can be run with a JIT compiler. This makes Kotlin applications just as portable and fast as any Java application.

Kotlin additionally uses the same garbage collector as Java, as it is run in the Java Virtual Machine and thus a programmer writing an application with Kotlin does not have to worry about manually freeing objects.

Finally, Kotlin is generally a statically typed language, meaning that types are checked during compile time. However, Kotlin has the additional advantage of its keyword dynamic. This keyword essentially disables Kotlin's static type checker.[11] This allows a programmer to have full flexibility, using the keyword dynamic when necessary but also generally having the advantages of a static type checker in every other scenario.

6.3 Drawbacks

One drawback of using Kotlin with TensorFlow is that Kotlin does not have bindings yet and is not does not support TensorFlow as well as Python does.

Additionally, because Kotlin is a newer language, there is less support for it in general. There are fewer resources for developers to refer to, such as StackOverflow, when they run into trouble or have any questions during the development of the application.

7 Conclusions

After discussing the benefits and drawbacks of three alternative languages, it seems that moving from using TensorFlow in Python to TensorFlow in one of the alternative languages could potentially solve the performance issues of our application. Due to the lack of adequate TensorFlow support in OCaml and Kotlin, in addition to the lack of portability of OCaml code, it seems that creating and testing a Java prototype would be beneficial, as Java is likely the most useful language to switch to in our application.

8 References

[1] *Getting Started with TensorFlow.* TensorFlow. Available: https://www.tensorflow.org/tutorials

[2] *TensorFlow in Other Languages.* TensorFlow. Available: https://www.tensorflow.org/guide/extend/bindings

[3] *Install TensorFlow for Java.* TensorFlow. Available: https://www.tensorflow.org/install/lang_java

[4] *TensorFlow Java API r1.13.* TensorFlow. Available: https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary

[5] *GlobalInterpreterLock.* Python. Available: https://wiki.python.org/moin/GlobalInterpreterLock

[6] *What is OCaml?* Ocaml. Available: https://ocaml.org/learn/description.html

[7] *TensorFlow bindings for OCaml.* Opam. Available: https://opam.ocaml.org/packages/tensorflow/

[8] *Multicore OCaml.* OCaml Labs. Available: http://oc
amllabs.io/doc/multicore.html

[9] *TensorFlow in Kotlin/Native.* Julius Kunze. January 24, 2018. Available: https://juliuskunze.com/tensorflow-in-kotlin-native.html

[10] *Comparison to Java Programming Language.* Kotlin. Available: https://kotlinlang.org/docs/reference/comparison-to-java.html

[11] *Dynamic Type.* Kotlin. Available: https://kotlinlang.org/docs/reference/dynamic-type.html

[12] *The Pros and Cons of Kotlin for Android Development.* DZone. Available: https://dzone.com/articles/the-pros-and-cons-of-kotlin-for-android-developmen