Homework 6: Tensorflow with Java, OCaml, and Kotlin

CS131 Student -- *University of California, Los Angeles*

Abstract

In this report, I will be analyzing the similarities and differences of three programming languages (Java, OCaml, and Kotlin) as well as weighing out their advantages and disadvantages in regards to their applicability for a prototype of an application server herd using Tensorflow for machine learning. We find that ultimately, the best language for our application is Kotlin due to its ease of use, efficiency, and high performance.

1 Introduction

Our goal is to run an application server proxy herd on a large set of virtual machines. Because the application uses machine learning algorithms, it relies very heavily on Tensorflow. A prototype has been built using Python that runs and could run well on large queries. Our goal is to speed up performance rate. To do this, we propose to possibly convert the application from Python code to a different language that may speed up performance. We would like to keep the use of Tensorflow and the possibility of prototyping with Python.

We will be analyzing the four possible languages available for our application, namely: Python, Java, OCaml, and Kotlin. In order to make the best decision, we will analyze the various different components of the languages such as accessibility, performance, generality, and efficiency. We more carefully would like to consider how the different aspects of each language will perform in when running servers that our application will include.

Tensorflow Overview

Tensorflow is an open source, cross platform library that is widely used for machine learning and eases the process of acquiring data, training models, serving predictions, and refining future results. It uses Python that provides a convenient API for building applications with the framework. It works by allowing developers to create dataflow graphs which are structures that show how data moves through a graph.

2 Language Comparisons

Python

Python is an imperative, object-oriented, and functional language that is popular due to its simplicity of use and applicability.

Python was the first client language that was supported by Tensorflow. Through Python, most features are supported. It is different from the Tensorflow C API in that the Python API has some extra features. Some of these include gradients, control flow, functions, and neural network libraries. Not only this, but the libraries all contain very thoroughly explained documentation which supports rapid development of an application that uses Tensorflow since the bindings are already written. This makes Python one of the best in terms of support for the use of Tensorflow.

Python is dynamically typed, which results in the fact that the variables in a program are not known until runtime. This may lead to less efficiency. In addition to being dynamically typed, it is strongly typed which means that there is a higher possibility of errors. Also, since it uses type inference, it infers the types of variables rather than having the programmer specify types of variables.

Python uses an interpreter rather than a compiler, which sacrifices speed since there is overhead from the translation process. This is not a problem in compiled languages.

In summary, one of the biggest advantages of Python is that there is a large amount of documentation and support for it. This can make it more convenient to

write applications since there is a large range of information available.

Java

Java is a language which is neither interpreted and it neither compiled. It uses a combination of the two. This method has certain advantages. For one, it is not as slow as Python. In regards to TensorFlow, like Python, Java has many resources. It is well supported and there are many online documents.

One of the advantages of using Java for Tensorflow is that the Java bindings that are implemented for Tensorflow use the development tool called Simplified Wrapper and Interface Generator which allows for the definition of a general purpose exception handler. In certain cases, Java's virtual machine can be put in danger when it runs native code. In this case, a runtime error can cause Java's virtual machine to crash and can break the application. This would not be a problem due to the bindings.

Another benefit of using Java is that it uses a static type system which guarantees that there will be reliability. Because Java is a statically typed language, the types are known during compilation. Because of this, the compiler can avoid certain semantic errors since it can check the behavior of objects.

However, the use of Java for this application could lead to some disadvantages. One of the disadvantages of Java is that there may be less portability available due to the way that its virtual machine executed the Java byte code. The portability is specifically lost when Java has to interface with C. Another disadvantage is that operator overloading is not possible in classes that are user-defined. This is a disadvantage in terms of flexibility.

OCaml

OCaml is a functional programming language that was initially developed to use for applications that involved symbolic computation such as automatic theorem provers, compilers, and program analyzers.

One of OCaml's strengths is that it offers a powerful type system. OCaml's type system supports parametric polymorphism and type inference. OCaml has the most powerful type inference than any other language. This allows for defining operations over a collection. Type inference also allows operations without having to worry about providing the type explicitly.
OCaml also features an automatic memory management due to a fast and incremental garbage collector.

However, OCaml does offer some disadvantages. For example, the fact that OCaml does not offer overleading can make numerical code over various different types more tedious. It also does not define a way to override equality, comparison, and hashign for new types in addition to not allowing the ability to catch errors in this context. This is caused because the default structural versions are not applicable. In this case, OCaml is not the best when trying to write code that is high performance and includes parallel and numerical code.

Kotlin

Kotlin is a general purpose statically typed programming language that combines object-oriented and functional programming features. It can be used in both styles, or the mixture of the two. It's focus is mainly on interoperability, clarity, safety, and tooling support. At a glance, it seems very similar to Java, in a more streamlined way.

Although its syntax is reminiscent of Java's, it is more concise. It is also more type safe than Java since it has support for non-nullable types which make applications less prone to NPEs. Other advantages include smart casting, higher order functions, and lambdas. On another note, Kotlin compiles to JVM bytecode or Javascript. Additionally, it imposes no runtime overhead. The standard library is small and consists mostly of focused extensions to the Java standard library. It is fully compatible with Java.

Coroutines available in Kotlin allow for the creation of multiple pieces of code that can run asynchronously. However, there are some things that are available in Java that Kotlin does not have. These include features such as checked exceptions, primitive types that are not classes, static members, non-private fields, wildcard-types, and ternary operators.

Conclusion

As one can see, these languages all have many similarities and differences, all which bring advantages and disadvantages. In our case, we would like to create an application server herd that will handle a large number of servers at a time. It is therefore important to use a language that can support this, is efficient, and will not create any unnecessary overhead. Because of this, I would recommend using Kotlin for our application. This is mainly due to the fact that out of all the sources consulted, the emphasis and selling point was that Kotlin will give you highly efficient implementations of patterns. It is also easy to use, and less code needs to be written. Due to this, it would benefit the application in that fewer performance degrading mistakes would occur.

Bibliography

[1] Alison DeNisco Rayome. "Why Kotlin is exploding in popularity among young developers". https://www.techrepublic.com/article/why-kotlin-is-exploding-in-popularity-among-young-developers/

[2] HOFFMANN, J., AND DAS, A. Towards automatic resource bound analysis for ocaml. cs.cmu.edu/~janh/papers/HoffmannW15.pdf.

[3] "Learn Kotlin". Kotlin, from https://kotlinlang.org/docs/reference/

[4] "What Is OCaml?" OCaml, ocaml.org/learn/description.html#Efficient-compiler-efficientcompiled-code.

[5] TensorFlow. "TensorFlow for Java." GitHub, github.com/tensorflow/tensorflow/tree/master/tensorflow/java.

[6] TensorFlow in Kotlin/Native. (January 2018). https://juliuskunze.com/tensorflow-in-kotlin-native.html