Scala Compiler Plugin for GPU support

Student

Reto Ehrensperger

Introduction: Graphics Processing Units (GPUs) are designed to process vast amounts of data in parallel, making them ideal for accelerating specific tasks. General-purpose graphics processors (GPGPUs) have extended the use of GPUs beyond graphics processing. However, existing GPU frameworks such as CUDA and OpenCL require developers to know about GPU design and only support C, C++, and Fortran without additional wrappers. Developers using programming languages other than the supported ones find it harder to accelerate their code on the GPU.

Approach: In this paper we propose a novel approach that enables Scala developers to accelerate parts of their code on the GPU without requiring knowledge of GPU design. We introduce a new compiler plugin that extends the abstract syntax tree of the code parts to be executed on the GPU. This plugin adds new nodes that enable execution on the GPU without needing low-level knowledge. Our approach provides an easy-to-use solution for Scala developers to take advantage of GPU acceleration without the need to learn additional frameworks or modify their existing code significantly.

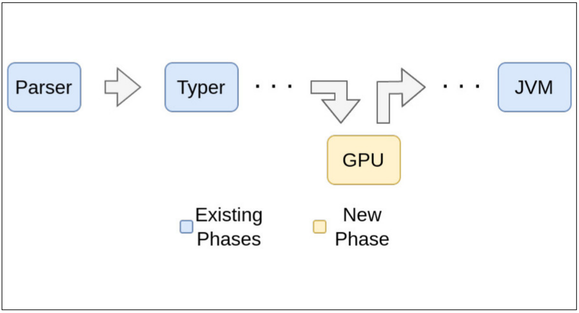
Conclusion: This project demonstrates the feasibility of using a compiler plugin for the Scala programming language to generate GPU kernels. Additionally, through our evaluation using the matrix multiplication algorithm, we have shown that using this compiler plugin can improve the performance of a Scala application when processing large data sets. While this compiler plugin has limitations, we believe it represents a promising approach to making GPU acceleration more accessible to developers in the Scala community.

A simple example of an array addition using the compiler plugin
Own presentment

```scala
object ArrayAddition {
  @Parallel
  def add(length: Int, a: Array[Int] ,b: Array[Int],
        c: Array[Int]): Unit = {
    for(i <- 0 parallelTo length) {
      c(i) = a(i) + b(i)
    }
  }

  def main(args: Array[String]): Unit = {
    ...
    add(c.length, a, b, c)
    ...
  }
}
```

Compiler procedure to generate the kernel code
Own presentment



Benchmark of multiplying a M x K matrix with a K x N matrix
Own presentment

M	N	K	CPU (ms)	GPU (ms)	Ratio
1000	1000	10	16	182	0.09
1000	1000	100	115	214	0.54
1000	1000	1000	2495	460	5.42
1000	1000	10000	65804	3320	19.82
100	100	100000	9118	511	17.84

Advisor
Prof. Dr. Mitra Purandare

Subject Area
Computer Science