Uncovering Nested Data Parallelism and Data Reuse in DNN Computation with FractalTensor

Summary

Extended Task Dependence Graph: is the main data structure that they use in to a achieve their speed up. Deep Nerual Networks depend on nested loops that iterate over data in the high deminsional array. They create a recusive data type called a FractalTensor. A FractalTensor can either be a static size tensor or a list of other FractalTensors objects. You can then operate on these data structures using map/reduce/scan operations. Extended Task Dependence Graph provides a holistic view of the parallelism

- Programming model: they create the Extended Task Dependence Graph (ETDG) data structure
- Dependence driven global analysis: in this stage they try to reduce the depth of the ETDG to increase the parallelism.
- Code emitter: the data structor is mapped on to an execution plan

There does not seem to be a connection to distributed systems. The only connection to distributed system I see is that the data types can be used in a distributed setting. They use distributed system models like parallelism and pipeline parallelism.

There is some conflict in the community between "expressiveness and efficiency" of DNN programming.

Pros

- the conclusion did a good job of clearing up some of my fundimental confuses from the abstract and introduction.
- the related works section would give some good pointers to do more research on DNN.

Cons

- I need a better understanding of DNN to understand this paper.

Further Developments

This seem like a very technology specific paper. Because it is a specific optimization for DNN it will be harder to find extentions.

Other Comments

What are tensor operations and why are they traditionally stored in a directed ascyclic graph? These questions come from the first sentence of the conclusion. I also wonder what they preciously mean by DNN programming.