Distillation of Sparse Linear Algebra

Aleksey Tyurin
St. Petersburg State University, Russia
JetBrains Research, Russia
alekseytyurinspb@gmail.com

Ekaterina Vinnik
St. Petersburg State University, Russia
JetBrains Research, Russia
catherine.vinnik@gmail.com

Mikhail Nikolukin
!!!
!!!

Daniil Berezun
St. Petersburg State University, Russia
JetBrains Research, Russia
d.berezun@2009.spbu.ru

Semyon Grigorev
St. Petersburg State University, Russia
JetBrains Research, Russia
s.v.grigoriev@spbu.ru
semyon.grigorev@jetbrains.com

Geoff Hamilton
School of Computing,
Dublin City University, Ireland
geoffrey.hamilton@dcu.ie

This is a sentence in the abstract. This is another sentence in the abstract. This is yet another sentence in the abstract. This is the final sentence in the abstract.

1 Introduction

Nowadays high-performance processing of a huge amount of data is an actual challenge not only for scientific computing but for applied systems. Special types of hardware, such as General Purpose Graphic Processing Units (GPGPUs), Tensor Processing Units (TPUs), FPGA-based solutions, along with respective specialized software were developed to provide appropriate solutions, and the development of new solutions continues.

. The simple well-known example is a pipelined processing of collections: `map g (map f data)`. Suppose `data` is a list, then the first map produces a new list which then will be traversed by the second map. It is important that not only list will be traversed twice while can be traversed only once, but the intermediate list will also be stored (to RAM) which is a big problem for real-world data analysis: the size of data is huge and memory access is one of the expensive operations. This case, and even more complex real-world cases, can be successfully optimized by deforestaion [**?**]. Stream fusion libraryes [**?**]

Deforestation, stream fusion (fold/unfold transformations). kernels fusion XLA [1], MLIR, sparse kernels fusion. But problem still not solved.

Linear algebra is a common language which brings together such areas as ML [3], graph analysis, graphblas(t) [2]. Machine learning, Social network analysis

Specialized hardware for functional languages.

2 Proposed Solution

We propose to ANYDSL [**?**] Futhark [**?**] We propose to use distillation [**?**] as an optimization method Tail-modulo-cons Multiply-add

General-purpose functional-language-friendly hardware: Reduceron [**?**] Specialization for Reduceron [**?**]

program-specific hardware FHW [**?**] FPGA. Kiwi

High-level view of architecture is provided in figure **??**

Submitted to:
VPT 2022

3 Preliminary Evaluation

Quad-tree for matrix representation. Basic functions: matrix-matrix elemetnwise operations, matrix-scalar elementwise operation, masking.

Few examples

4 Future Work

In the future, first of all, we should close a technical debt and make the distiller more stable to handle all important cases: current implementation can not handle such important functions as matrix-matrix multiplication. Along with it, we should improve the input language to make it more user-friendly. The main challenge here is to find the balance between language expressivity and the practicality of distillation for it.

When the language and the distiller will be stable enough, we plan to implement a full-featured generic linear algebra library power enough to express basic graph analysis algorithms and to create and train neural networks. After that, a number of graph analysis algorithms and neural networks will be implemented and evaluated.

Along with it we plan to improve both FHW and Reduceron and compilers for it in order to make them mature enough to handle real-world examples. For example, it is necessary to support out-of-chip memory.

References

[1] Todd Wang Chris Leary (2017): *XLA: TensorFlow, compiled.* `https://developers.googleblog.com/2017/03/xla-tensorflow-compiled.html`.

[2] Carl Yang, Aydin Buluç & John D. Owens (2019): *GraphBLAST: A High-Performance Linear Algebra-based Graph Framework on the GPU*. *CoRR* abs/1908.01407. arXiv:1908.01407.

[3] Zhen Zheng, Pengzhan Zhao, Guoping Long, Feiwen Zhu, Kai Zhu, Wenyi Zhao, Lansong Diao, Jun Yang & Wei Lin (2020): *FusionStitching: Boosting Memory Intensive Computations for Deep Learning Workloads*. *CoRR* abs/2009.10924. arXiv:2009.10924.