Distilling Sparse Linear Algebra

Aleksey Tyurin[1,2]
alekseytyurinspb@gmail.com
[1] *JetBrains Research, Russia* [2] *St. Petersburg University, Russia*

Problem Statement

Fusion is an ubiquitous optimization for dence applications widely used, e.g., in Tensorflow, aimed to reduce memory usage. Is is also a highly desired optimization in sparse applications [1] but it is hard to implement due to pointer-chaising [2] nature of the latter. The basics of this optimization is the removal of *intermediate* data structures : those which are firstly constructed and then deconstructed. This is common for functional programming where such optimization is often addressed as *deforestation.*

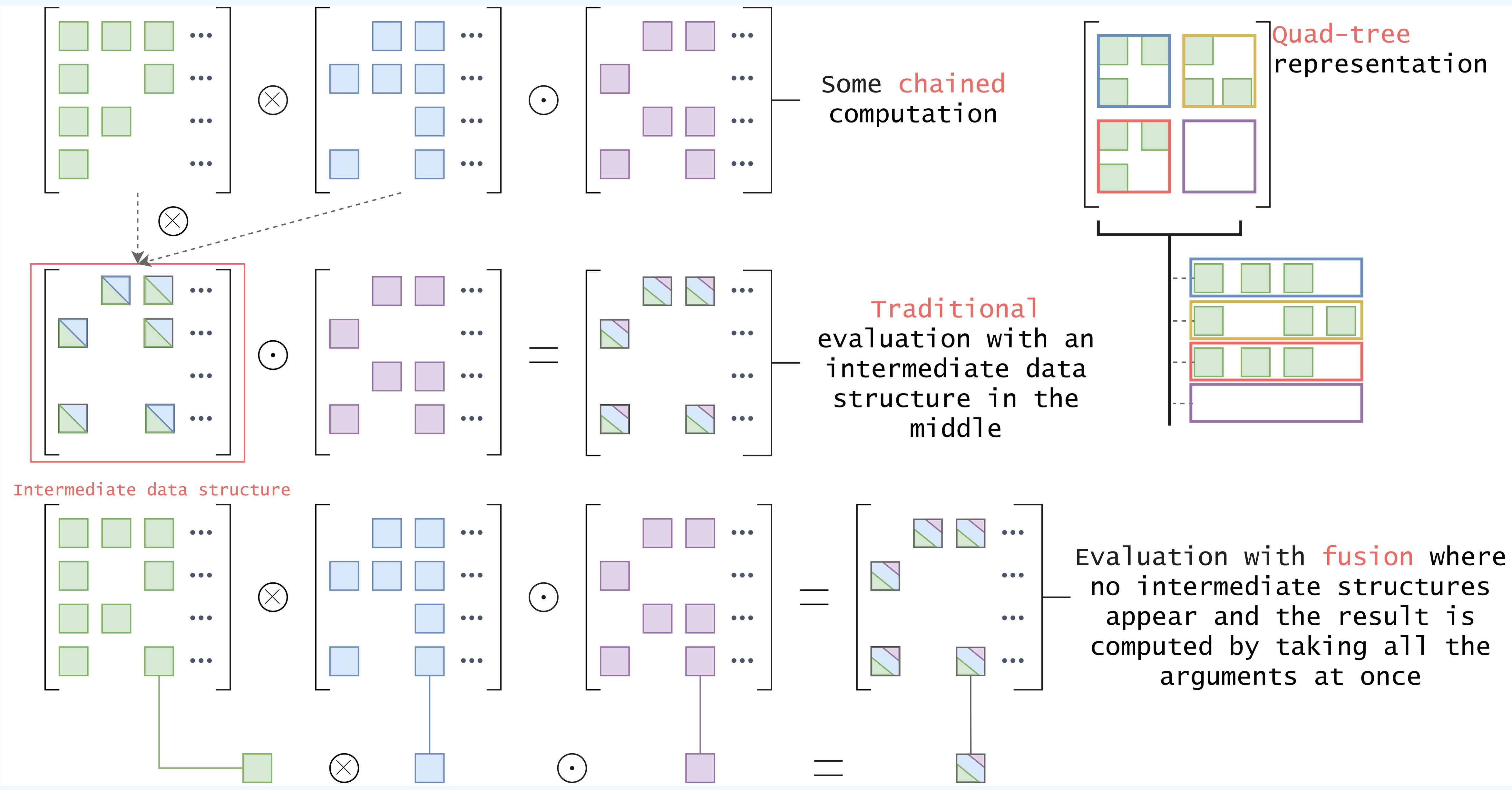
We propose the usage of a functional *quad-tree* represenation for sparse data and *disillation* to support fusion for sparse applications.

Distillation

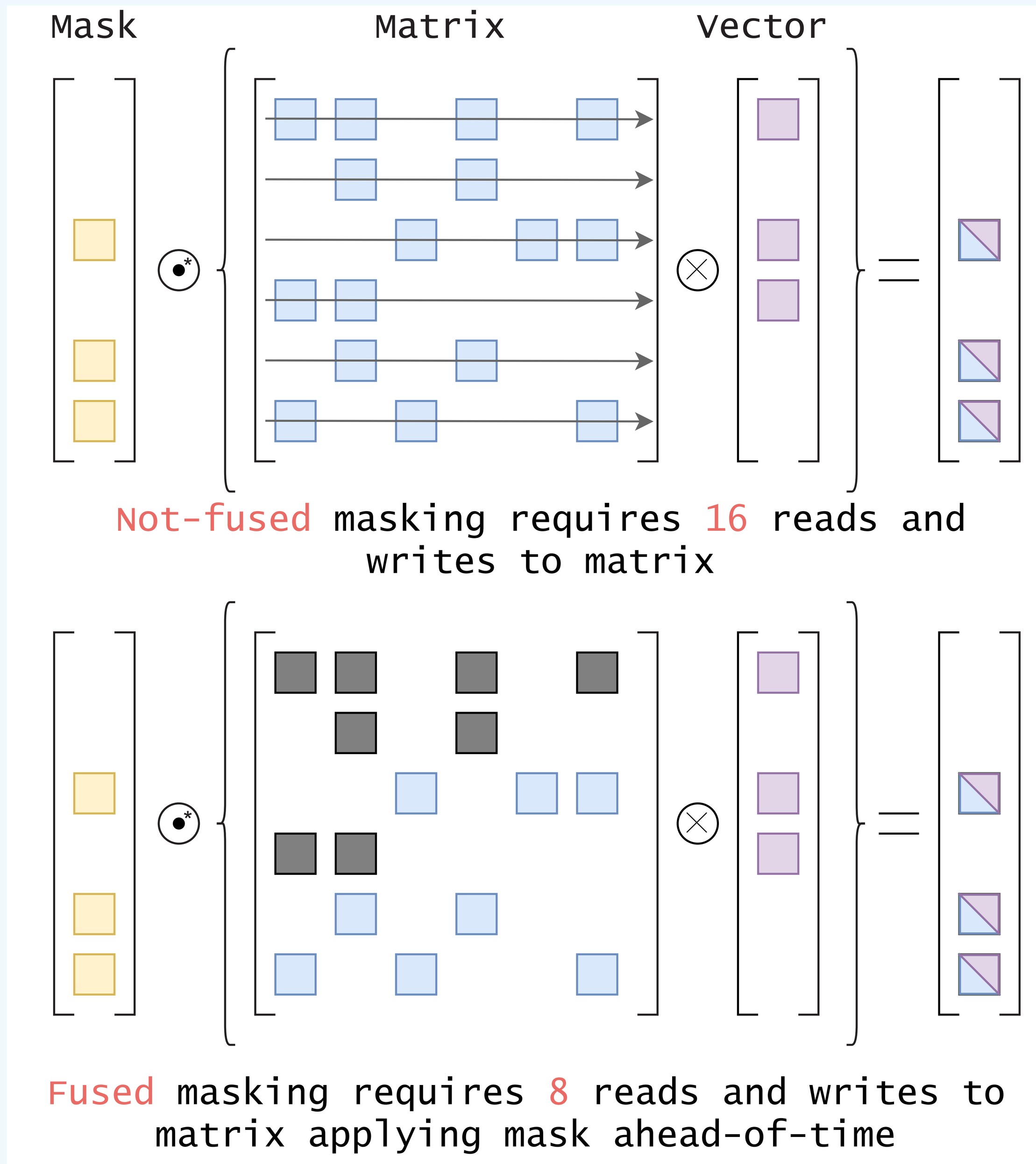
Distillation implements deforestation by removing intermediate data structures, i.e. those first constructed and then deconstructed, providing the following bonuses

- Specialization, i.e., it partially evaluates the program on statically known arguments.
- Yields *tail recursive modulo cons* programs, which could ease the following translation to hardware.
- Gives potentially assympotically greater speed-up than deforestation.

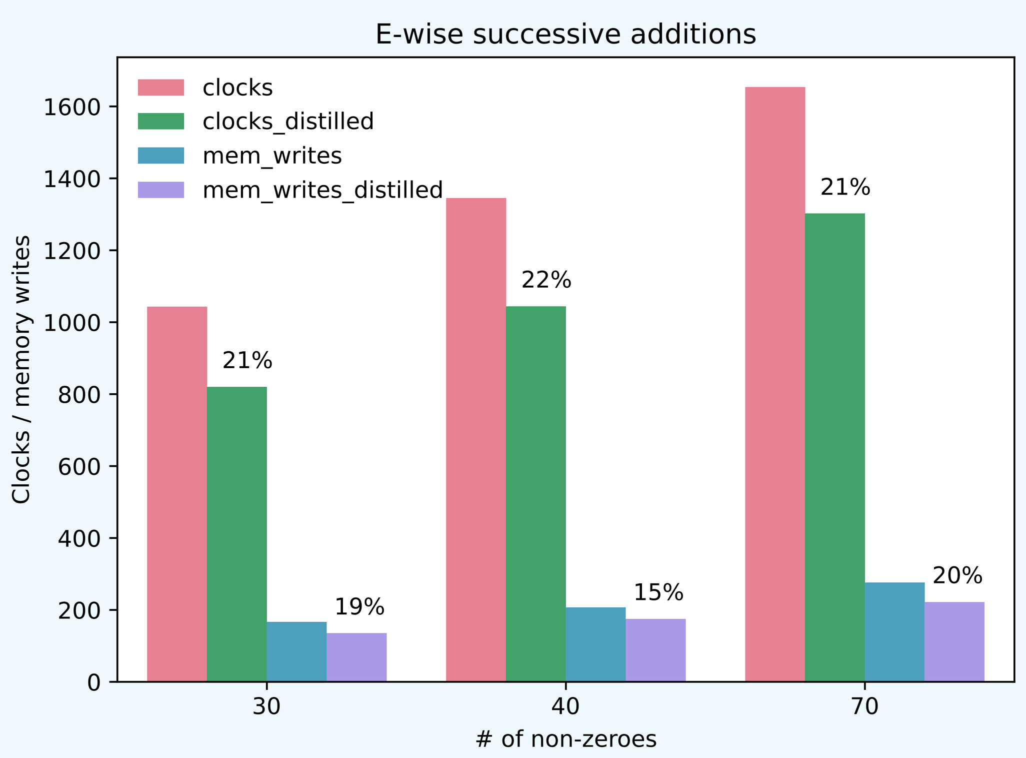
Motivation: Kernel fusion



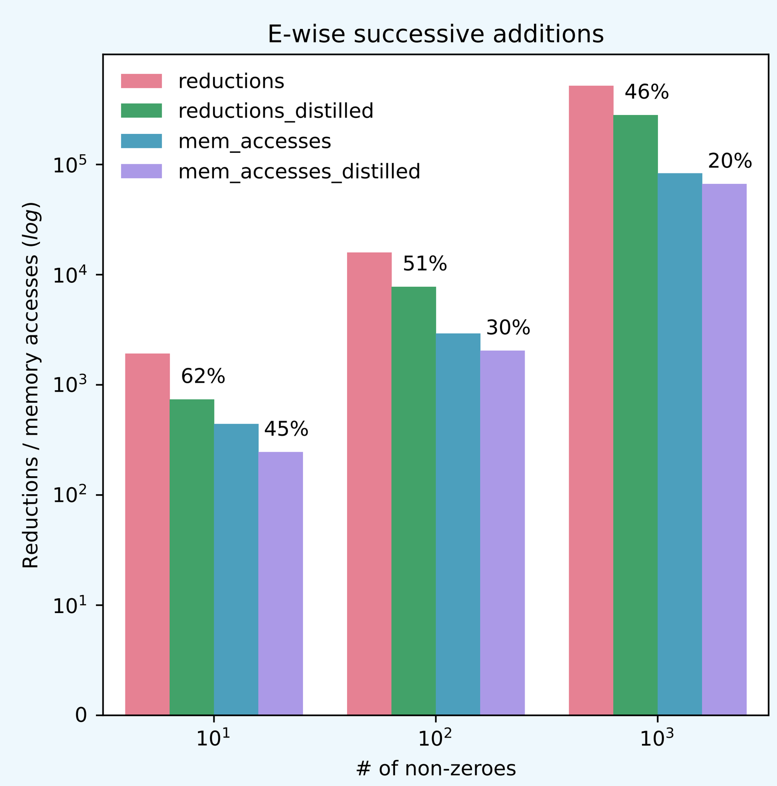
Motivation: Mask fusion [1]

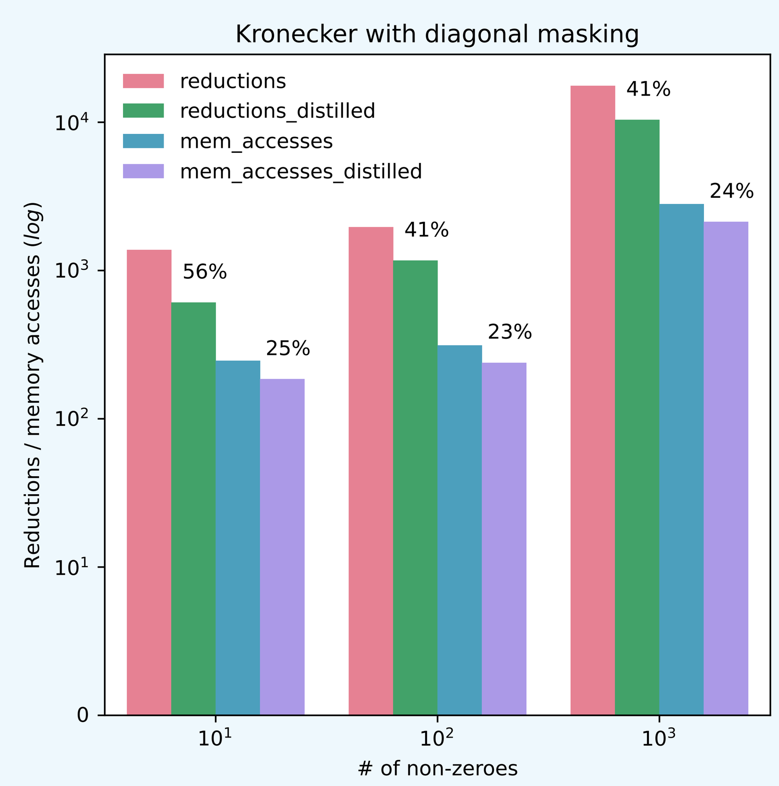


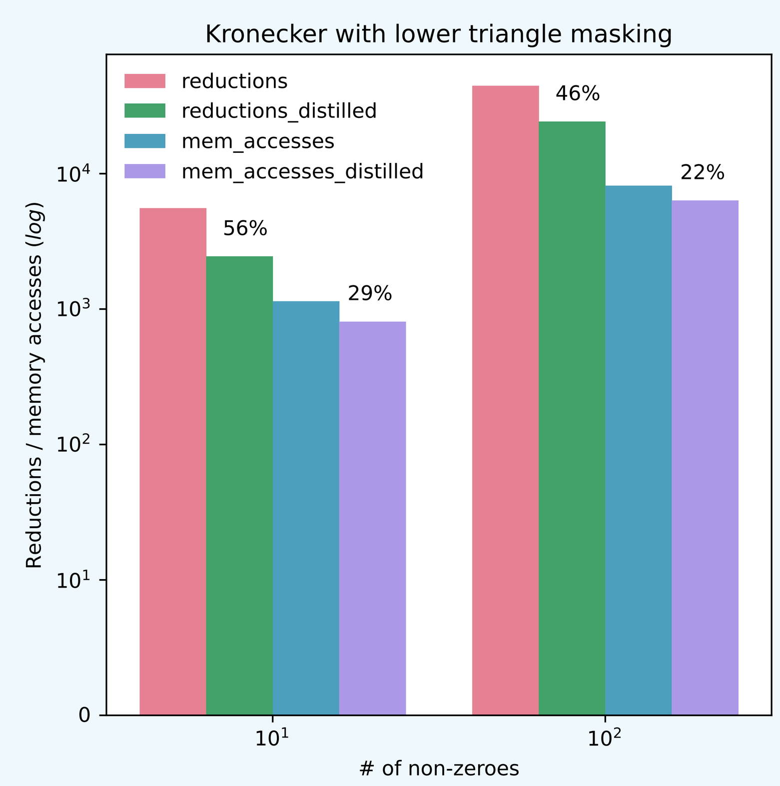
Evaluation: Hardware



Evaluation: Software







Implementation

We use the distiller authored by Geoff Hamilton [3] and its functional language to evaluate the approach in terms of *reductions* and *memory accesses.*
In order to provide both enough performance and interoperability with C++ (in which modern sparse frameworks are mostly written) we aim to synthesize a FPGA kernel from distilled functional program and utilize FHW project [4] to do so.

Results

Distillation gives prominent results namely

- Shows up to **60%** less reductions and **45%** less memory accesses in software.
- Shows up to **20%** less clock cycles and memory writes in hardware.

Future Research

- Moving the disiller from proof-of-concept to ready-to-use.
- Moving the hardware compiler from proof-of-concept to ready-to-use.
- Bridge the gap between our approach and existing sparse frameworks in a form of OpenCL-like kernels.
- Real-world examples evaluation.

Contact Us

Our team:

- Aleksey Tyurin: alekseytyurinspb@gmail.com
- Daniil Berezun: daniil.berezun@jetbrains.com
- Ekaterina Vinnik: catherine.vinnik@gmail.com
- Semyon Grigorev: s.v.grigoriev@spbu.ru

References

[1] Carl Yang, Aydin Buluc, and John D. Owens. Graphblast: A high-performance linear algebra-based graph framework on the gpu, 2020.

[2] Troels Henriksen, Niels G. W. Serup, Martin Elsman, Fritz Henglein, and Cosmin E. Oancea. Futhark: Purely functional gpu-programming with nested parallelism and in-place array updates. *SIGPLAN Not.*, 52(6):556–571, June 2017.

[3] Geoff Hamilton. Extracting the essence of distillation. pages 151–164, 06 2009.

[4] S. Edwards, Martha A. Kim, Richard Townsend, Kuangya Zhai, and L. Lairmore. Fhw project : High-level hardware synthesis from haskell programs. 2019.