How to Implement Generic Matrix-Mul with Generic Element Types on GPU?

Xuanzhao Gao

The Hong Kong University of Science and Technology (Guangzhou)

Acknowledgement

Instructor: Jin-Guo Liu
HKUST(GZ)

OSPP 2023

Julia CN

CONTENTS

- How to implement generic element type in Julia?
- Generic matrix multiplication (GEMM) on GPU.

Implementation of Generic Element Type in Julia

Using Tropical Numbers as an Example

What is Tropical Algebra?

- Tropical algebra is a set of semiring algebra, which is a generalization of a ring, dropping the requirement that each element must have an additive inverse.
- A tropical algebra can be defined by the set, add/mul rule and add/mul identity.

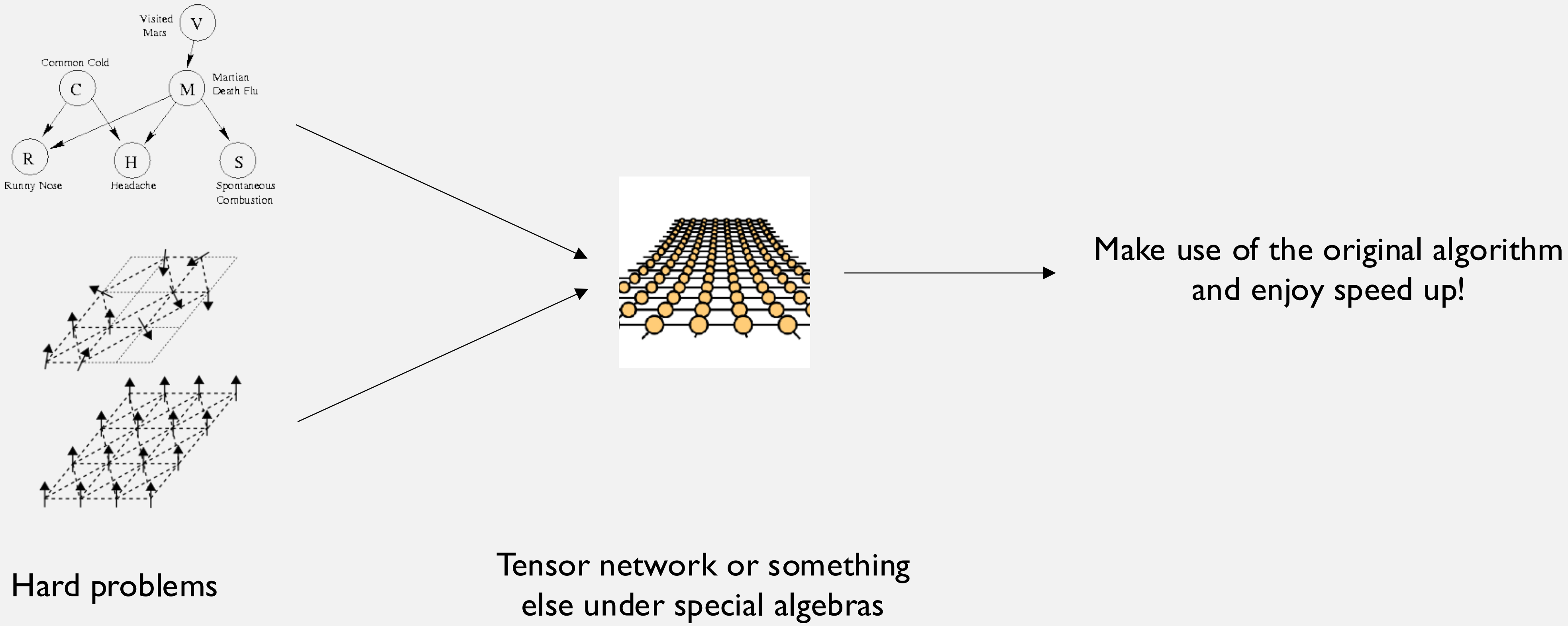
Algebra	set	Add-rule	Mul-rule	Add-identity	Mul-identity
TropicalAndOr	$[T, F]$	$\vee$	$\wedge$	F	T
TropicalMaxPlus	$\mathbb{R}$	max	$+$	$-\infty$	0
TropicalMinPlus	$\mathbb{R}$	min	$+$	∞	0
TropicalMaxMul	$\mathbb{R}^+$	max	$\times$	0	1

Why we need Tropical Algebra (or other generic algebra)?

- In recent years, the tropical numbers have been widely used in various areas, including optimization, physics, and computer science, due to its computational simplicity.
  - Ground state of spin glass system.
  - Probabilistic Inference (TensorInference.jl).
  - Semiring backpropagation.

Generalizing Backpropagation for Gradient-Based Interpretability, arXiv:2307.03056
Tropical Tensor Network for Ground States of Spin Glasses, 10.1103/PhysRevLett.126.090506

Why we need Tropical Algebra (or other generic algebra)?



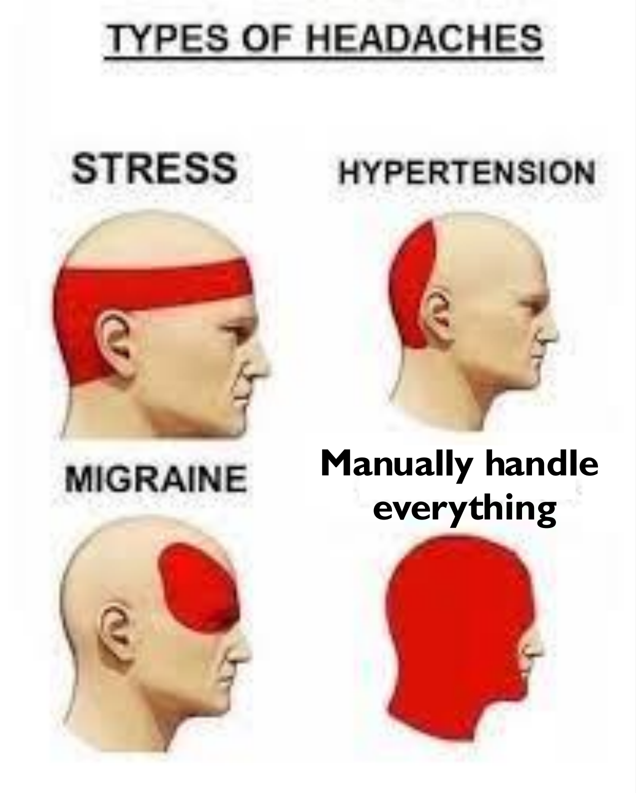
Hard problems

Tensor network or something else under special algebras

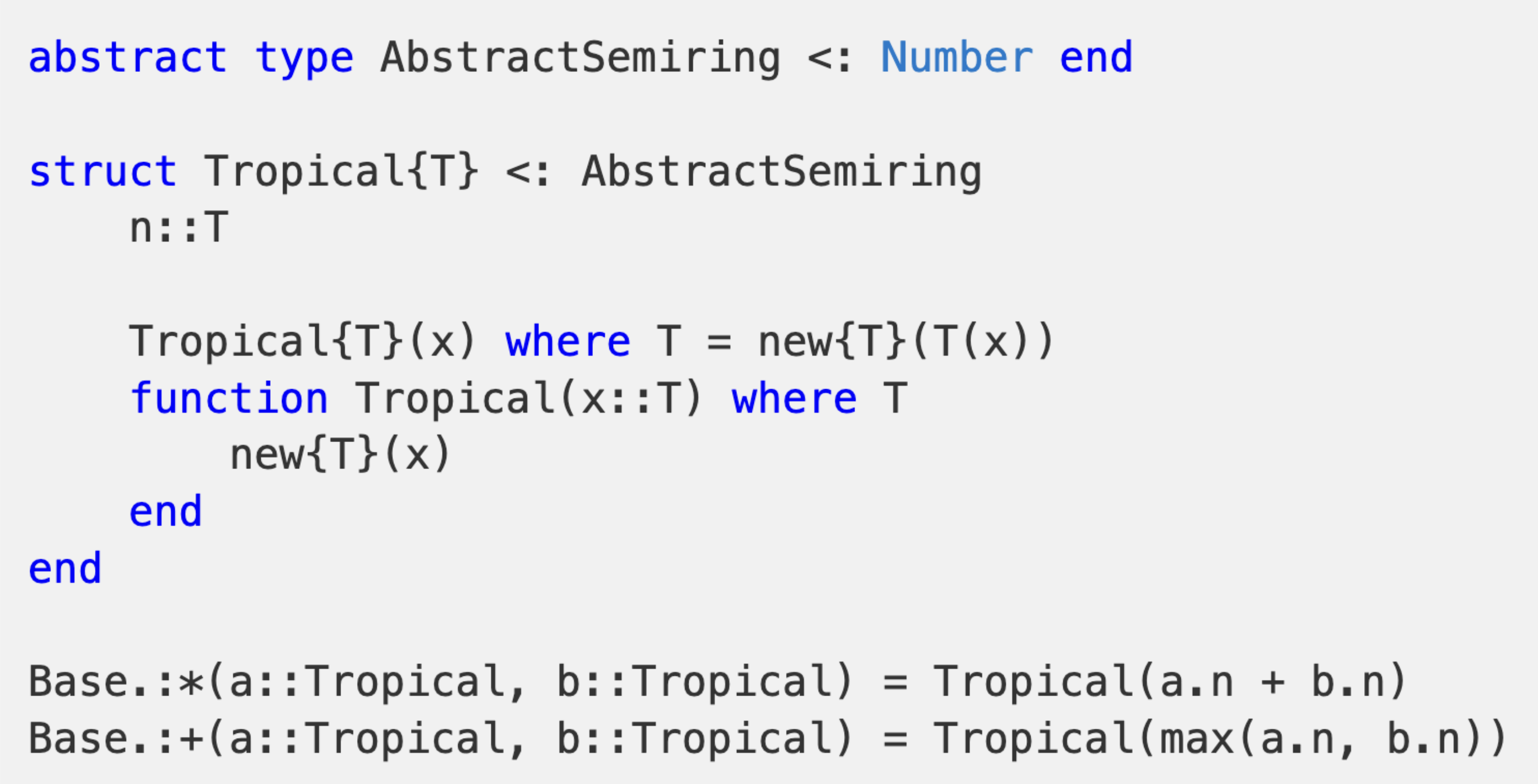
Make use of the original algorithm and enjoy speed up!

Naïve Approach

- What about handle everything manually?
- For example, you can use Float32 or Float64 as basic element type and change every + and × in your code manually.
- Of course this will be something really terrible.



Make use of multiple dispatch in Julia



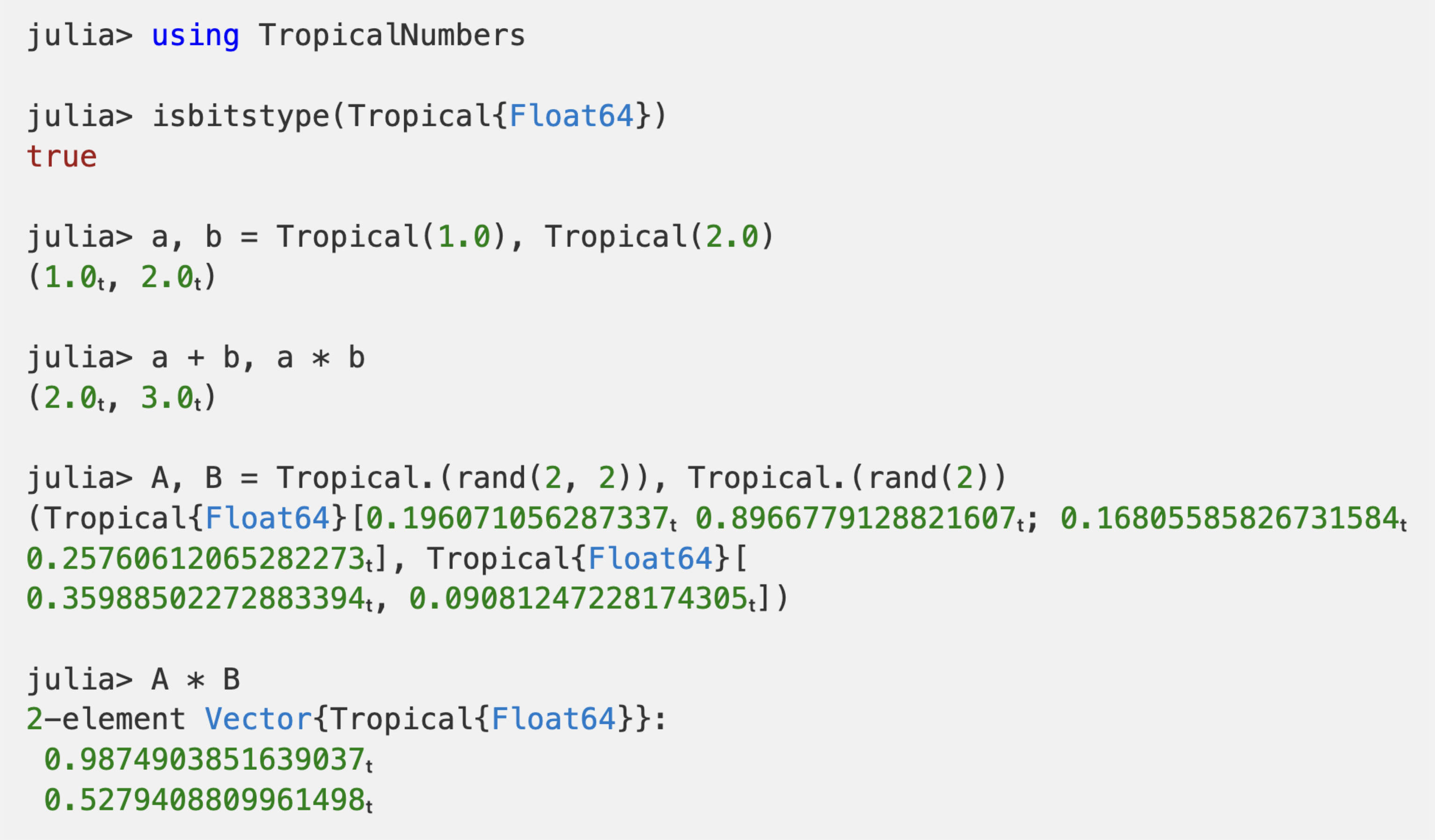
```julia
abstract type AbstractSemiring <: Number end

struct Tropical{T} <: AbstractSemiring
    n::T

    Tropical{T}(x) where T = new{T}(T(x))
    function Tropical(x::T) where T
        new{T}(x)
    end
end

Base.:*(a::Tropical, b::Tropical) = Tropical(a.n + b.n)
Base.:+(a::Tropical, b::Tropical) = Tropical(max(a.n, b.n))
```

Example of usage



```
julia> using TropicalNumbers

julia> isbitstype(Tropical{Float64})
true

julia> a, b = Tropical(1.0), Tropical(2.0)
(1.0ₜ, 2.0ₜ)

julia> a + b, a * b
(2.0ₜ, 3.0ₜ)

julia> A, B = Tropical.(rand(2, 2)), Tropical.(rand(2))
(Tropical{Float64}[0.196071056287337ₜ 0.8966779128821607ₜ; 0.16805585826731584ₜ
0.25760612065282273ₜ], Tropical{Float64}[
0.35988502272883394ₜ, 0.09081247228174305ₜ])

julia> A * B
2-element Vector{Tropical{Float64}}:
 0.9874903851639037ₜ
 0.5279408809961498ₜ
```

GEMM for Generic Element Type on GPU

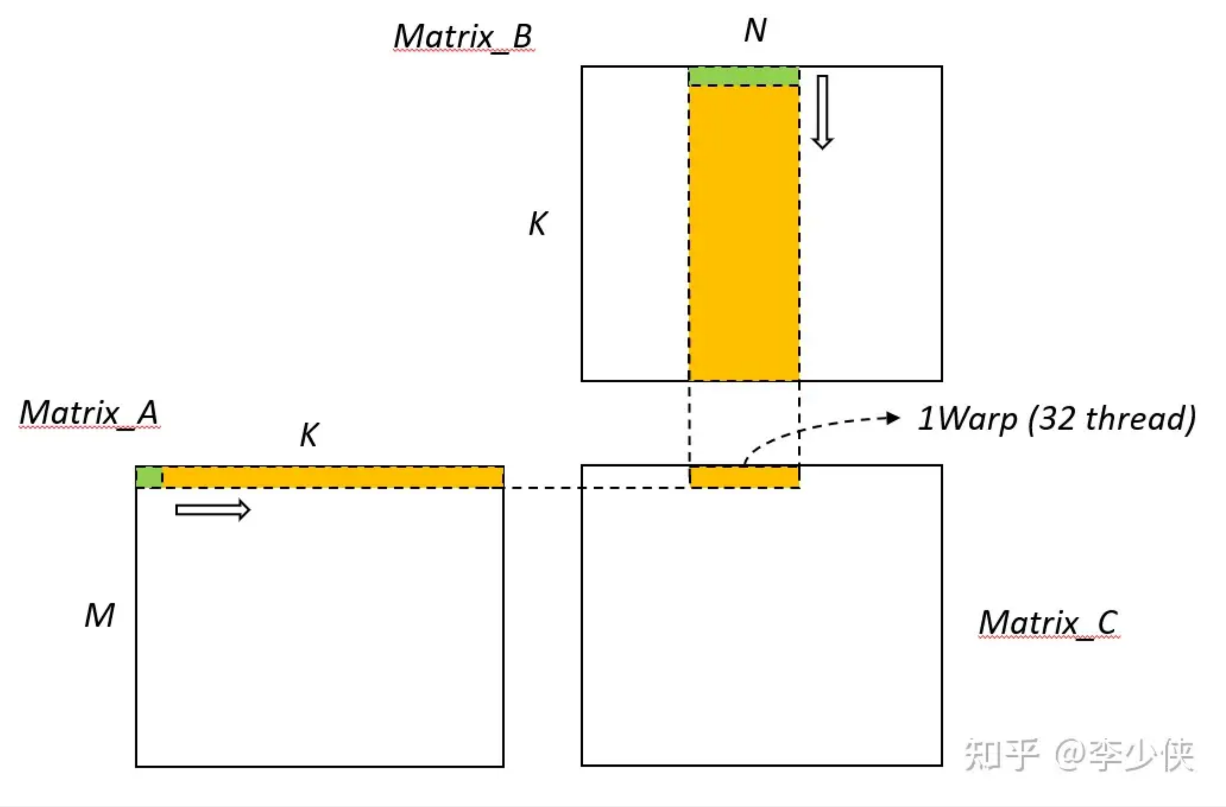
Again, using Tropical Numbers as an Example

Why is GPU fast?

- There are much more works in GPU that CPU. For example, the Nvidia A800 GPU have 6912 cuda cores, which is much larger than the number of CPU cores, which is normally less than 100.
- The peak flops of A800 (TDP 250W) is about 19.5T, which is also much higher than that of CPU, for example 0.5T for an Intel i7-10700K (125W).
- It is necessary to make all SM work together to reach a high performance.

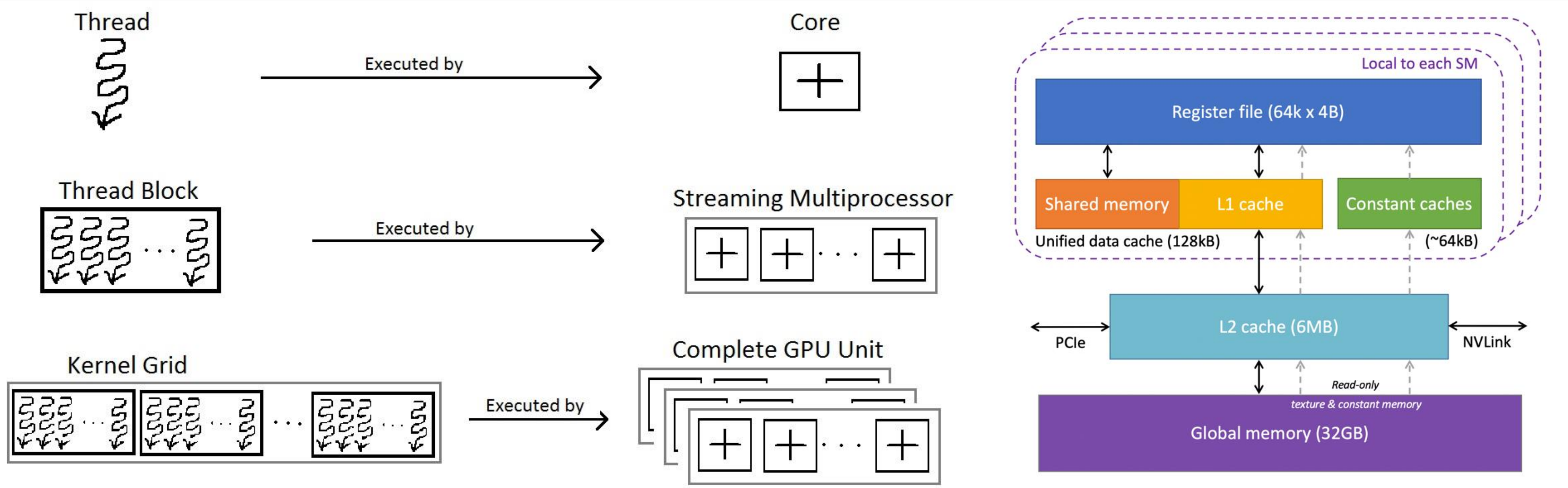
- How to use GPU for GEMM?

Naïve Approach



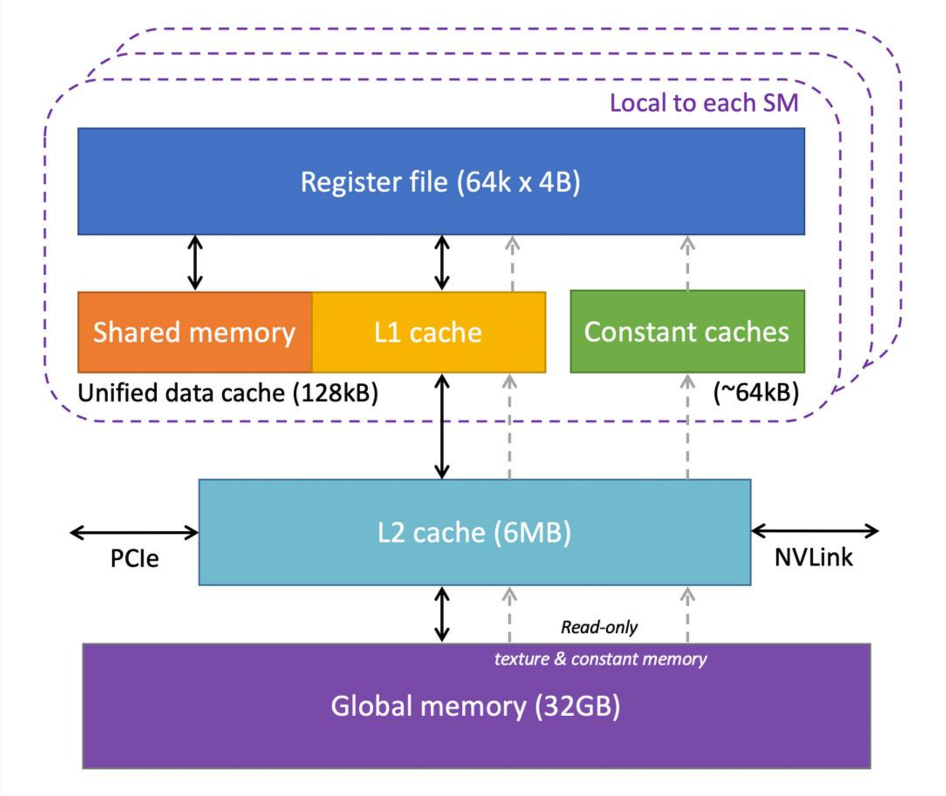
Will that work?

Structure of GPU



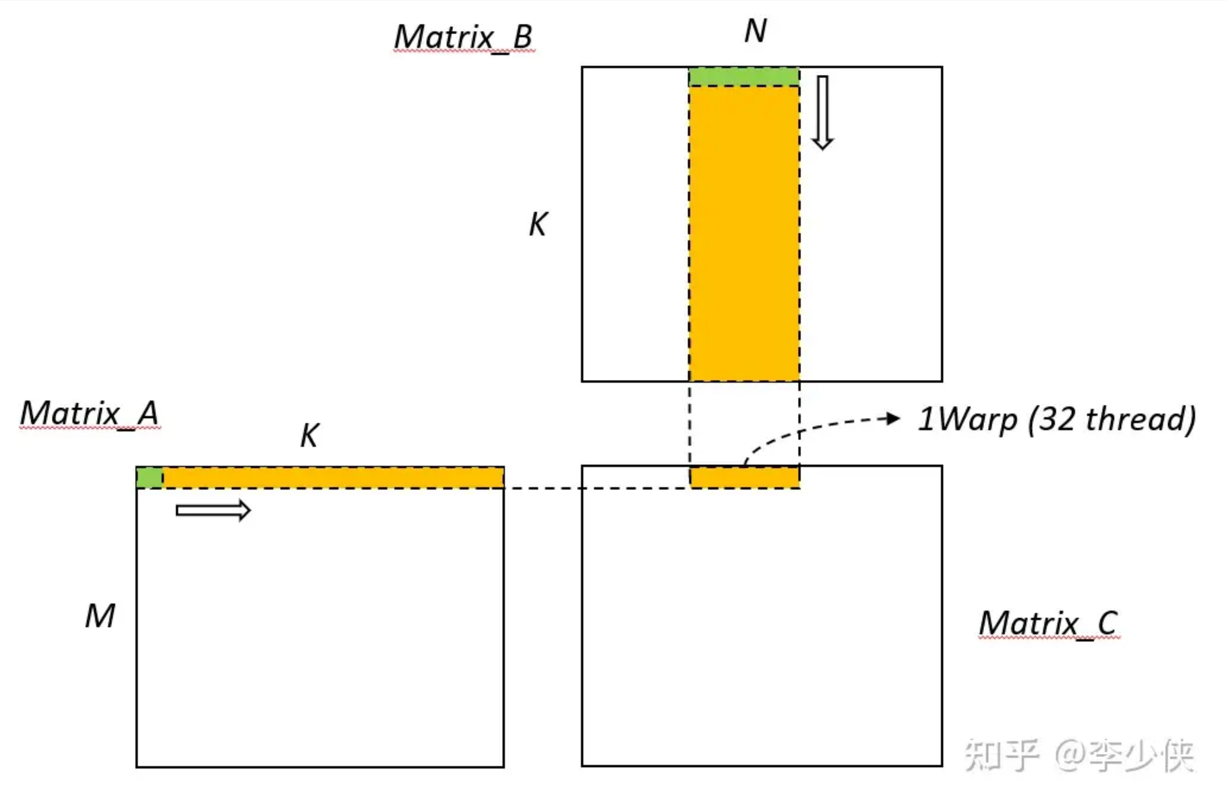
GPU memory levels and sizes for the NVIDIA Tesla V100.

Structure of GPU



Nvidia A800	Bandwidth	Latency
dram	1000 GB/s	585 cycles
L2	3235 GB/s	328 cycles
L1	N/A	33 cycles
Smem	19491 GB/s = 128 byte/cycle	23 cycles

Naïve Approach

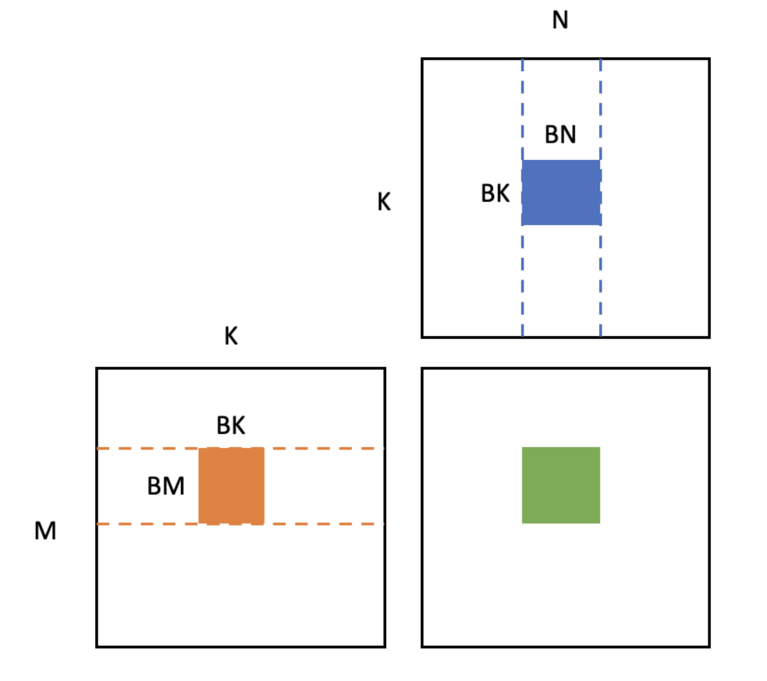


Operation/load = 64OP / 132byte = 0.48

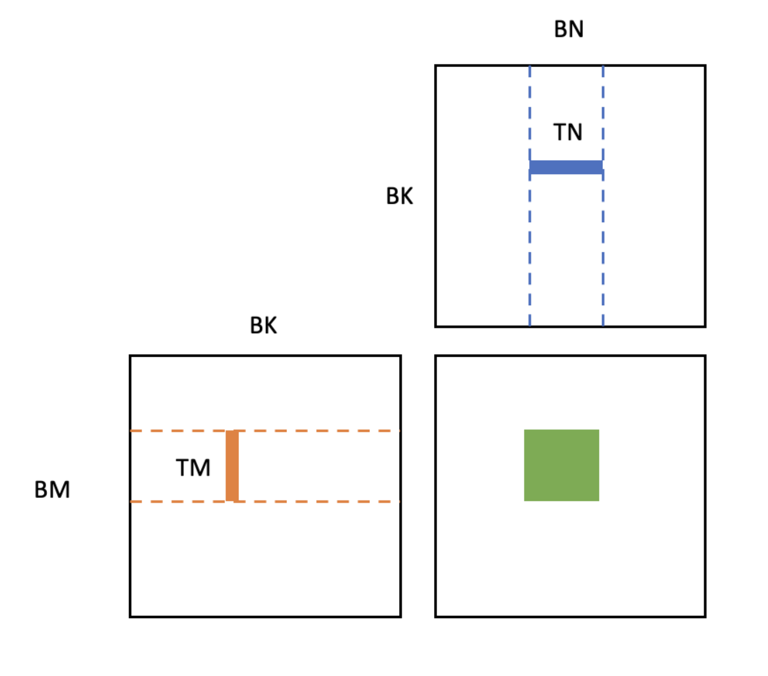
Bandwith of L2 Cache of A800: 3TB/s

Max performance ~ 1.5 Tflops << 19.5 Tflops

Tiling Algorithm



for blocks



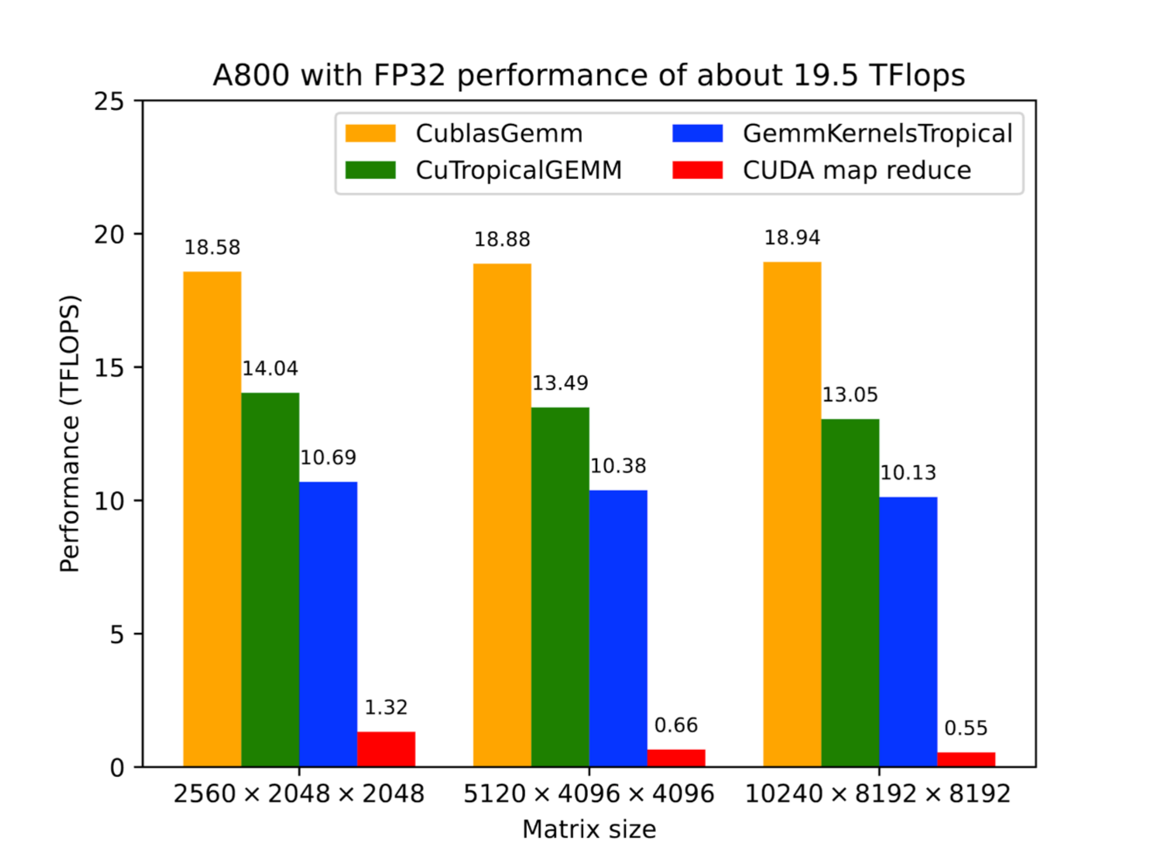
for threads

Operation/load = BM * BN / 2(BM + BN)

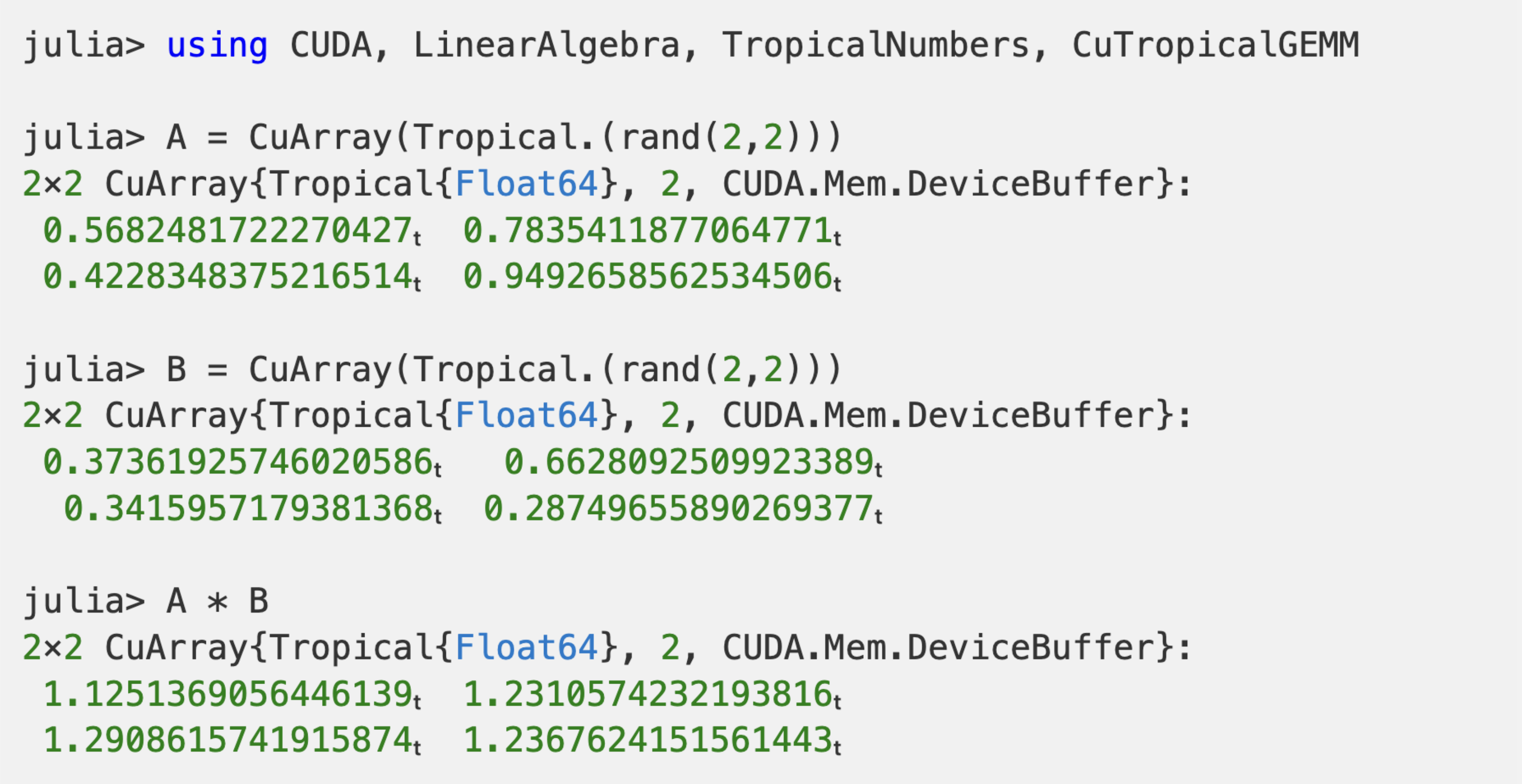
BM = BN = 64, Operation/load = 16

3TB/s * 16 = 48 Tflops >> 19.5 Tflops

Benchmarks


A800 with FP32 performance of about 19.5 TFlops
Performance (TFLOPS)
25
20
15
10
5
0
CublasGemm
CuTropicalGEMM
GemmKernelsTropical
CUDA map reduce
18.58
14.04
10.69
1.32
18.88
13.49
10.38
0.66
18.94
13.05
10.13
0.55
2560 × 2048 × 2048
5120 × 4096 × 4096
10240 × 8192 × 8192
Matrix size

Example of usage



```
julia> using CUDA, LinearAlgebra, TropicalNumbers, CuTropicalGEMM

julia> A = CuArray(Tropical.(rand(2,2)))
2×2 CuArray{Tropical{Float64}, 2, CUDA.Mem.DeviceBuffer}:
 0.5682481722270427ₜ  0.7835411877064771ₜ
 0.4228348375216514ₜ  0.9492658562534506ₜ

julia> B = CuArray(Tropical.(rand(2,2)))
2×2 CuArray{Tropical{Float64}, 2, CUDA.Mem.DeviceBuffer}:
 0.37361925746020586ₜ   0.6628092509923389ₜ
  0.3415957179381368ₜ  0.28749655890269377ₜ

julia> A * B
2×2 CuArray{Tropical{Float64}, 2, CUDA.Mem.DeviceBuffer}:
 1.1251369056446139ₜ  1.2310574232193816ₜ
 1.2908615741915874ₜ  1.2367624151561443ₜ
```

SUMMARY

- Implementation of generic element types in Julia
- Implementation of GEMM on GPU

TensorBFS/TropicalNumbers.jl

TensorBFS/CuTropicalGEMM.jl

Thanks!