FACULTY OF ENGINEERING
AND ARCHITECTURE

DEPARTMENT ELECTRONICS AND INFORMATION SYSTEMS
COMPUTER SYSTEMS LAB

PROGRAMMING NVIDIA GPUS WITH CUDANATIVE.JL

Tim Besard – 2017-06-21

GHENT
UNIVERSITY

TABLE OF CONTENTS

GHENT
UNIVERSITY

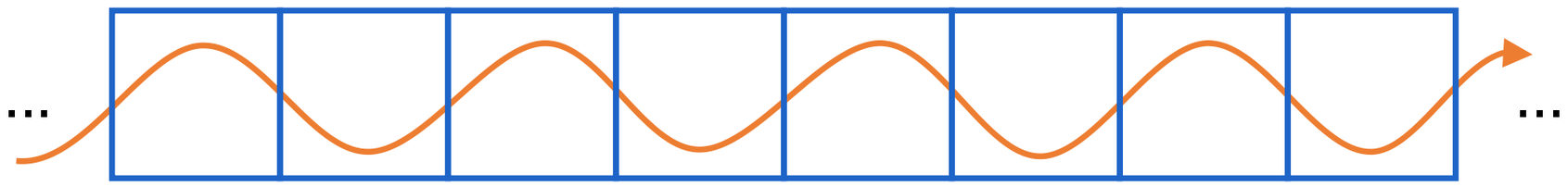
DISCLAIMER

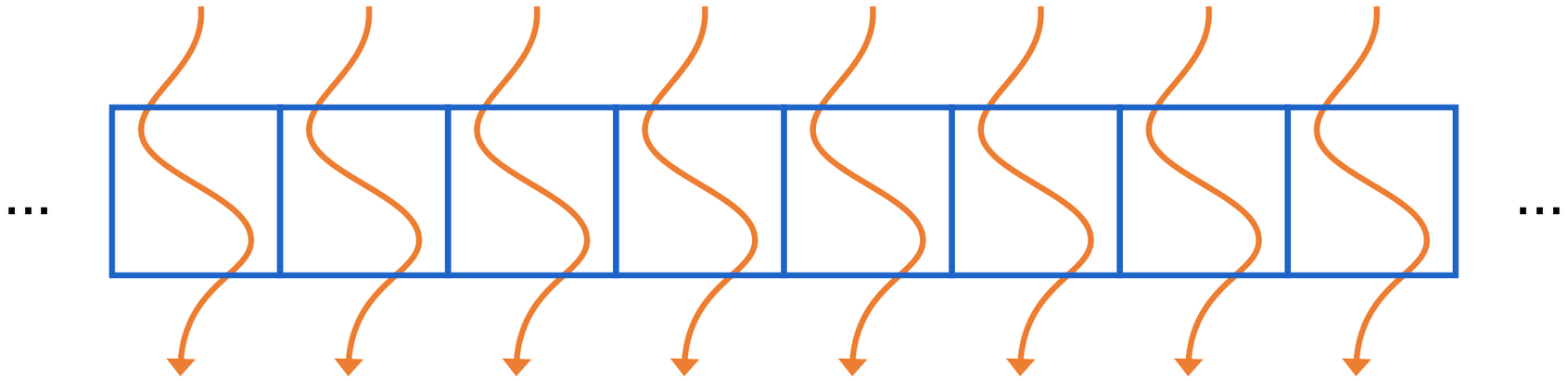
GPU PROGRAMMING:

WHAT, WHY AND HOW?

GHENT
UNIVERSITY

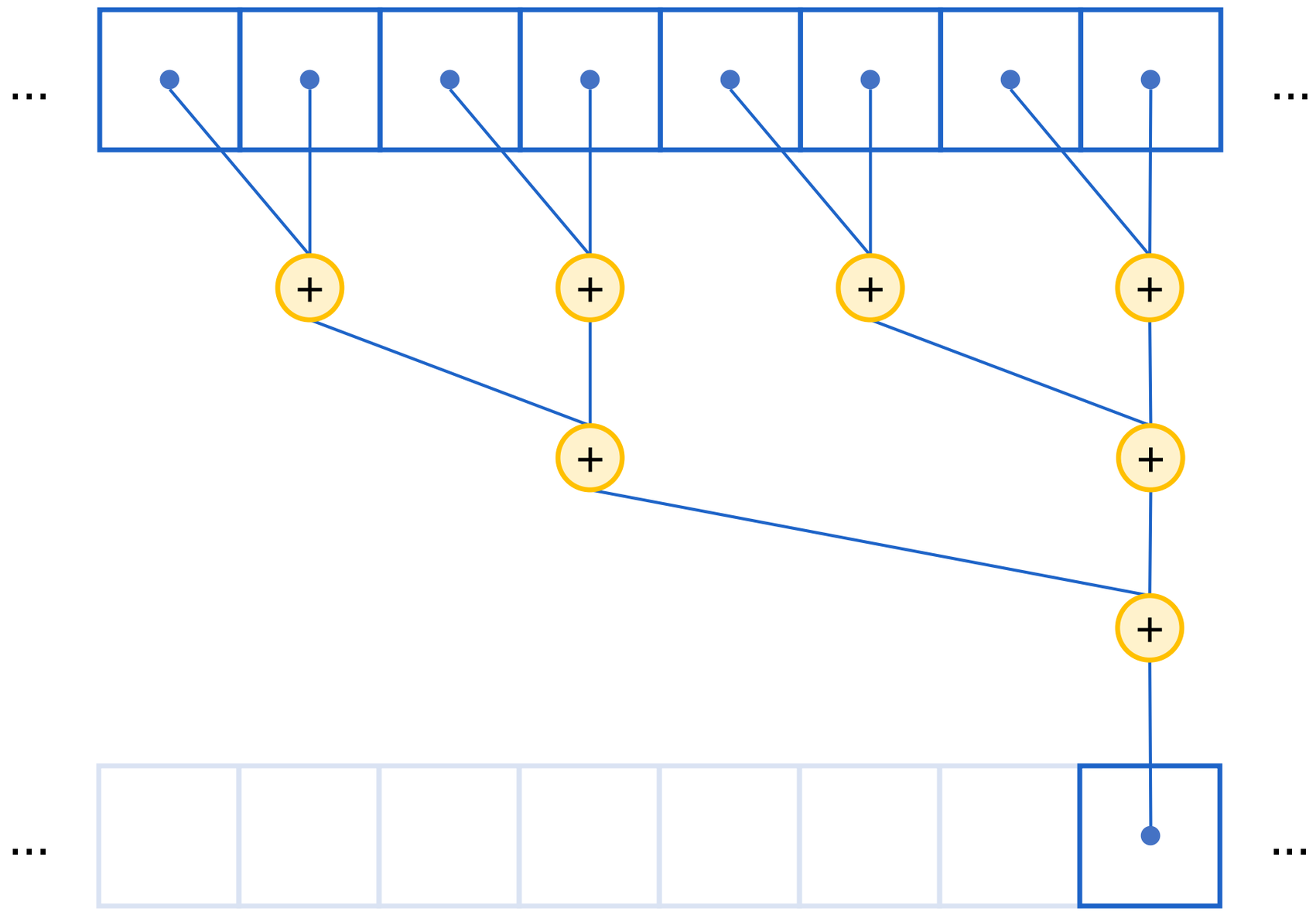
MASSIVE PARALLELISM





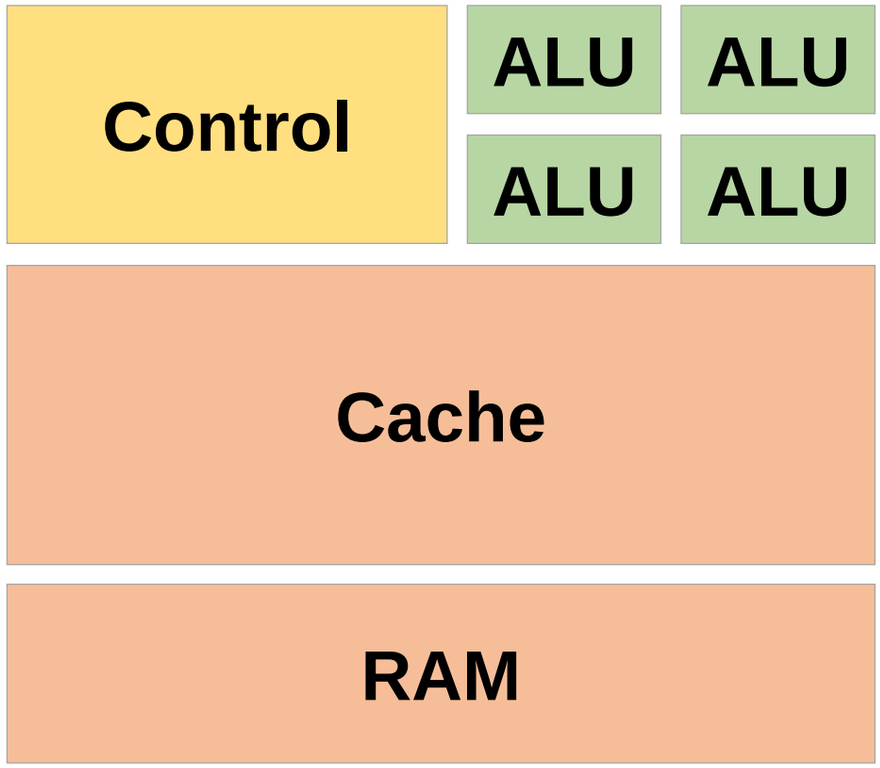
GHENT UNIVERSITY

PARALLEL SUM

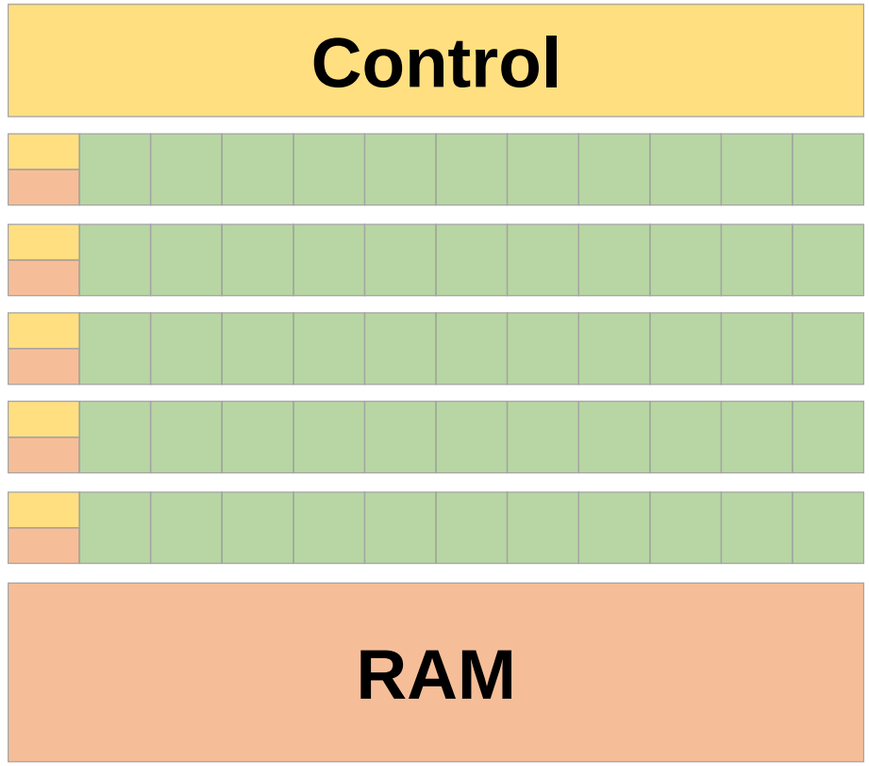


GHENT UNIVERSITY

ARCHITECTURE



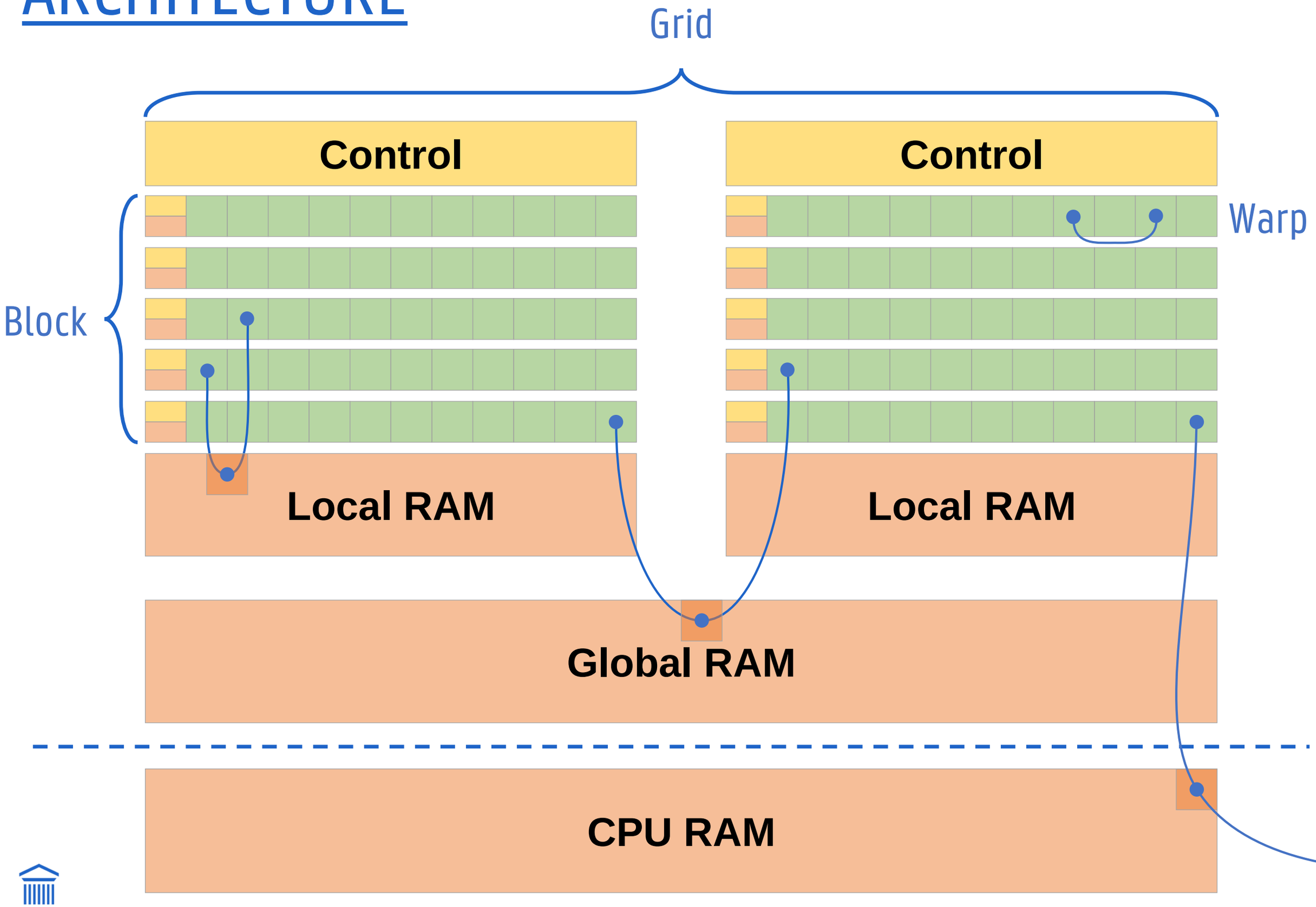
CPU



GPU

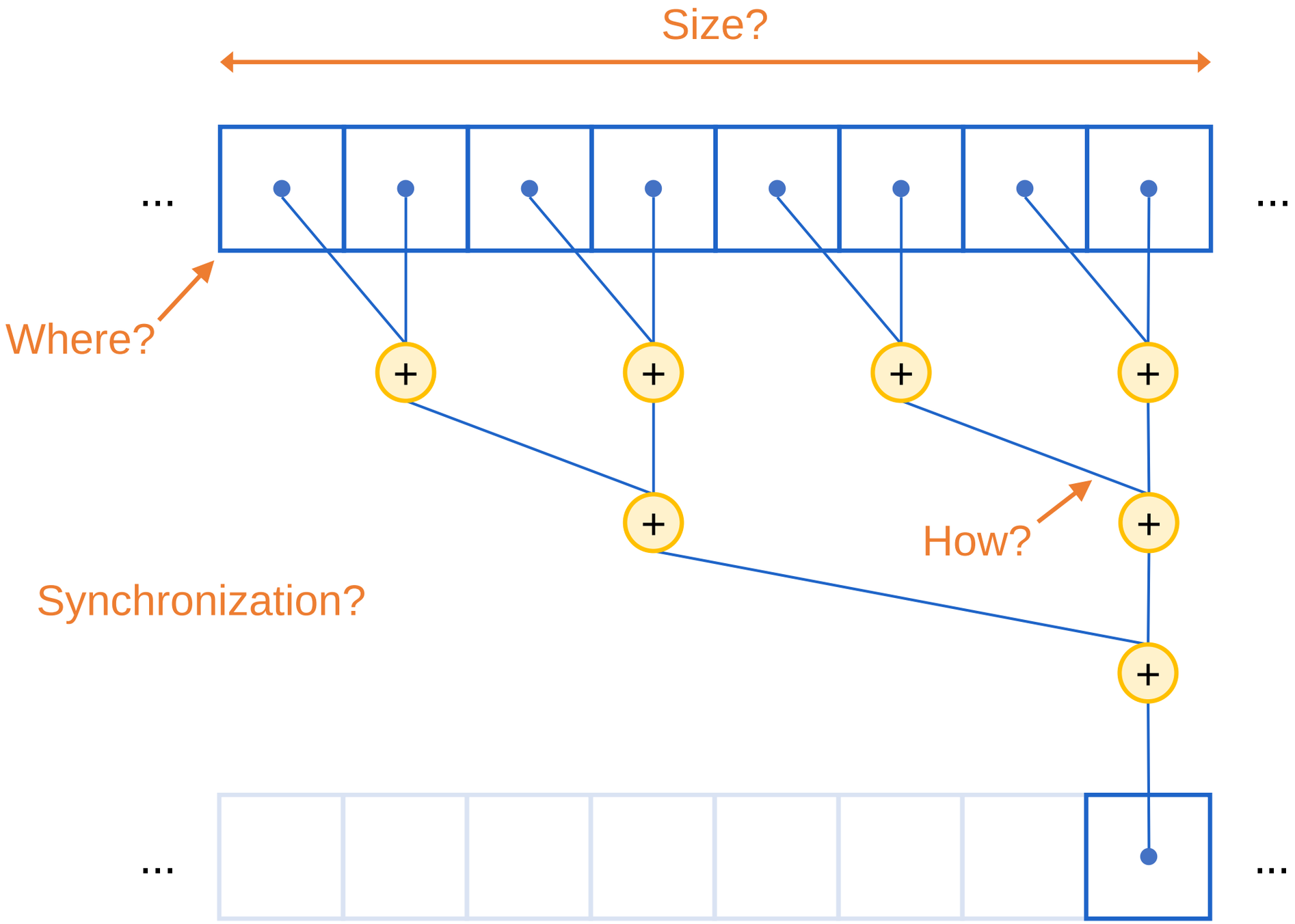
GHENT UNIVERSITY

ARCHITECTURE

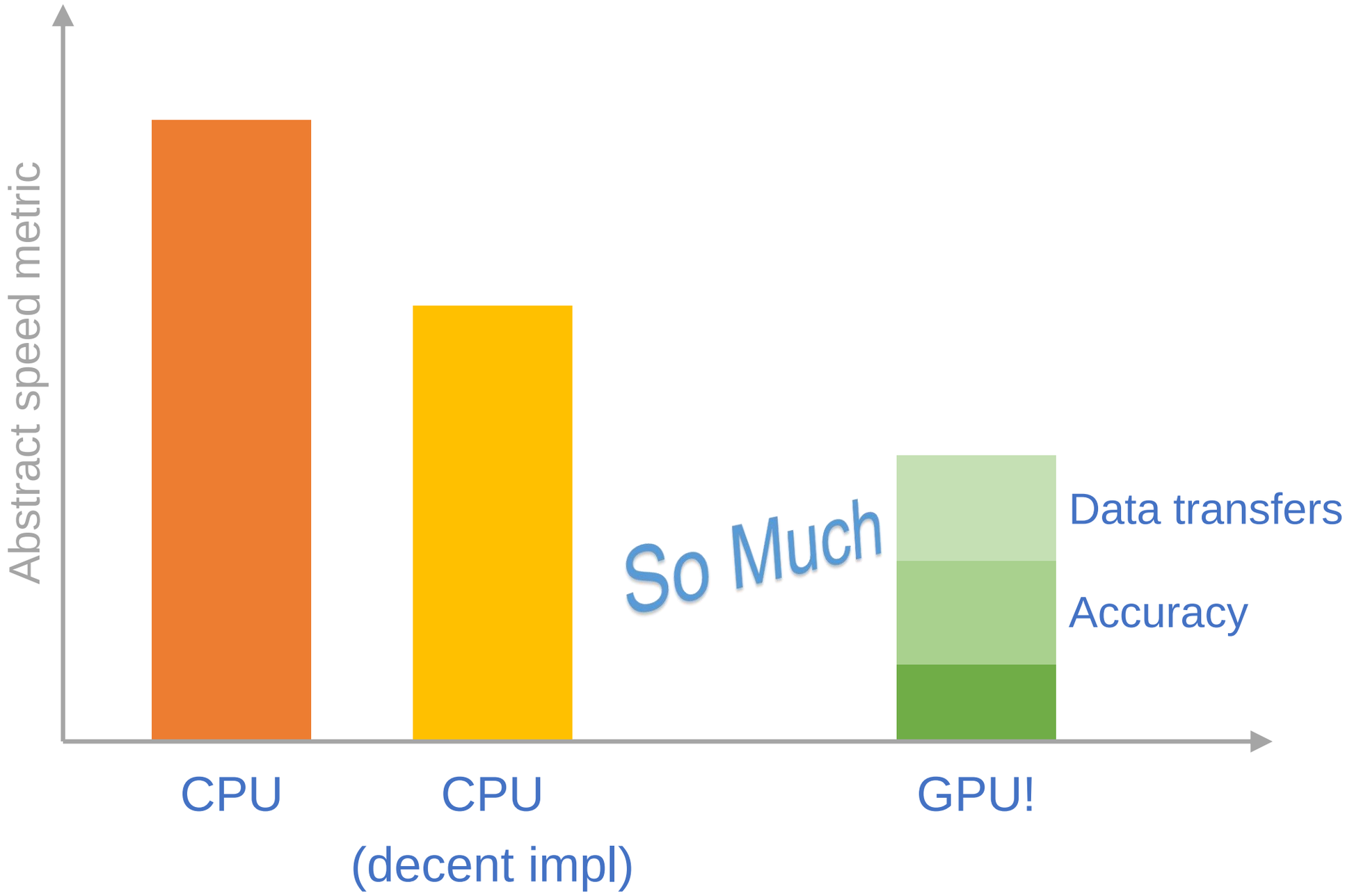


GHENT UNIVERSITY

PARALLEL SUM



WHY?



GHENT UNIVERSITY

HOW?

Transparant	Host libraries	Device code
TensorFlow, MXNet	cuBLAS, cuFFT, ... Thrust, CUB ArrayFire	CUDA rt + CUDA C CUDA rt + CUB

HOW?

Implemented using

Transparant	Host libraries	Device code
TensorFlow, MXNet	cuBLAS, cuFFT, ... Thrust, CUB ArrayFire	CUDA rt + CUDA C CUDA rt + CUB

Host libraries	Device code
cuBLAS.jl, cuFFT.jl, ... ArrayFire.jl	CUDArt.jl + CUDA C

CUDANATIVE.JL

Goal: replace CUDA C with Julia

- ✓ intrinsics
- ✓ code generation
- ✓ language integration

GHENT UNIVERSITY

CUDANATIVE.JL

Goal: replace CUDA C

Non-goal: make CUDA fun & easy

- ✗ automatic data management
- ✗ high-level parallelism
- ✗ portability

GHENT UNIVERSITY

CUDANATIVE.JL IN ACTION

GHENT
UNIVERSITY

QUICK START

```
Pkg.add("CUDAnative")
```

GHENT
UNIVERSITY

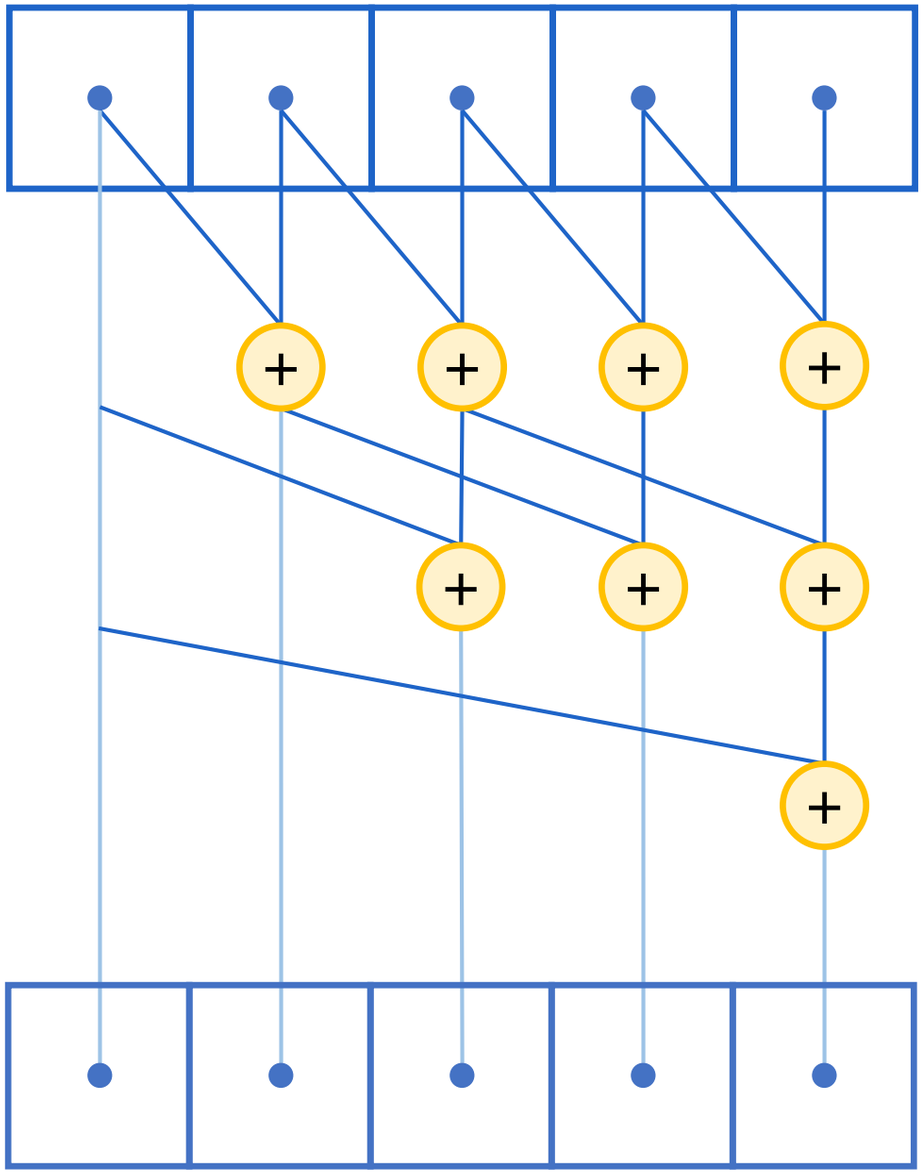
QUICK START: VADD

```
using CUDAdrv, CUDAnative

dev = CuDevice(0)
ctx = CuContext(dev)

len = 42
a = rand(Float32, len)
b = rand(Float32, len)

d_a = CuArray(a)
d_b = CuArray(b)
d_c = similar(d_a)
```

```
function vadd(a, b, c)
    i = threadIdx().x
    c[i] = a[i] + b[i]
    return
end

@cuda (1,len) vadd(d_a, d_b, d_c)

c = Array(d_c)
@test a+b ≈ c

destroy!(ctx)
```

GHENT UNIVERSITY

PREFIX SUM



```
function sum!(data)
    i = threadIdx().x

    offset = 1
    while offset < i
        a = data[i]
        b = data[i - offset]
        sync_threads()

        data[i] = a + b
        sync_threads()

        offset *= 2
    end
    return
end

@cuda (1,length(gpu_data))
      reduce!(gpu_data)
```

GHENT UNIVERSITY

PREFIX SUM

```
shmem = @cuStaticSharedMem(Float32, rows)
shmem[row] = data[row]

offset = 1
while offset < row
    sync_threads()
    a = shmem[row]
    b = shmem[row - offset]

    sync_threads()
    shmem[row] = a+b

    offset *= 2
end

sync_threads()
data[row] = shmem[row]
```

```
function sum!(data)
    i = threadIdx().x

    offset = 1
    while offset < i
        a = data[i]
        b = data[i - offset]
        sync_threads()

        data[i] = a + b
        sync_threads()

        offset *= 2
    end
    return
end
```

GHENT UNIVERSITY

PREFIX SUM

```
shmem = @cuStaticSharedMem(Float32, rows)
shmem[row] = data[row]

offset = 1
while offset < row
    sync_threads()
    a = shmem[row]
    b = shmem[row - offset]

    sync_threads()
    shmem[row] = a+b

    offset *= 2
end

sync_threads()
data[row] = shmem[row]
```

- Large arrays
- Intra-warp communication
- Optimize memory accesses

GHENT UNIVERSITY

CUDA C SUPPORT

- ✓ Indexing
- ✓ Synchronization
- ✓ Shared memory types
- ✓ Warp voting & shuffle
- ✓ Formatted output
- ✓ `libdevice`

GHENT UNIVERSITY

CUDA C SUPPORT

- Atomics
- Dynamic parallelism
- Advanced memory types

GHENT UNIVERSITY

JULIA SUPPORT

- `libjulia`
- Dynamic allocations
- Exceptions
- Recursion

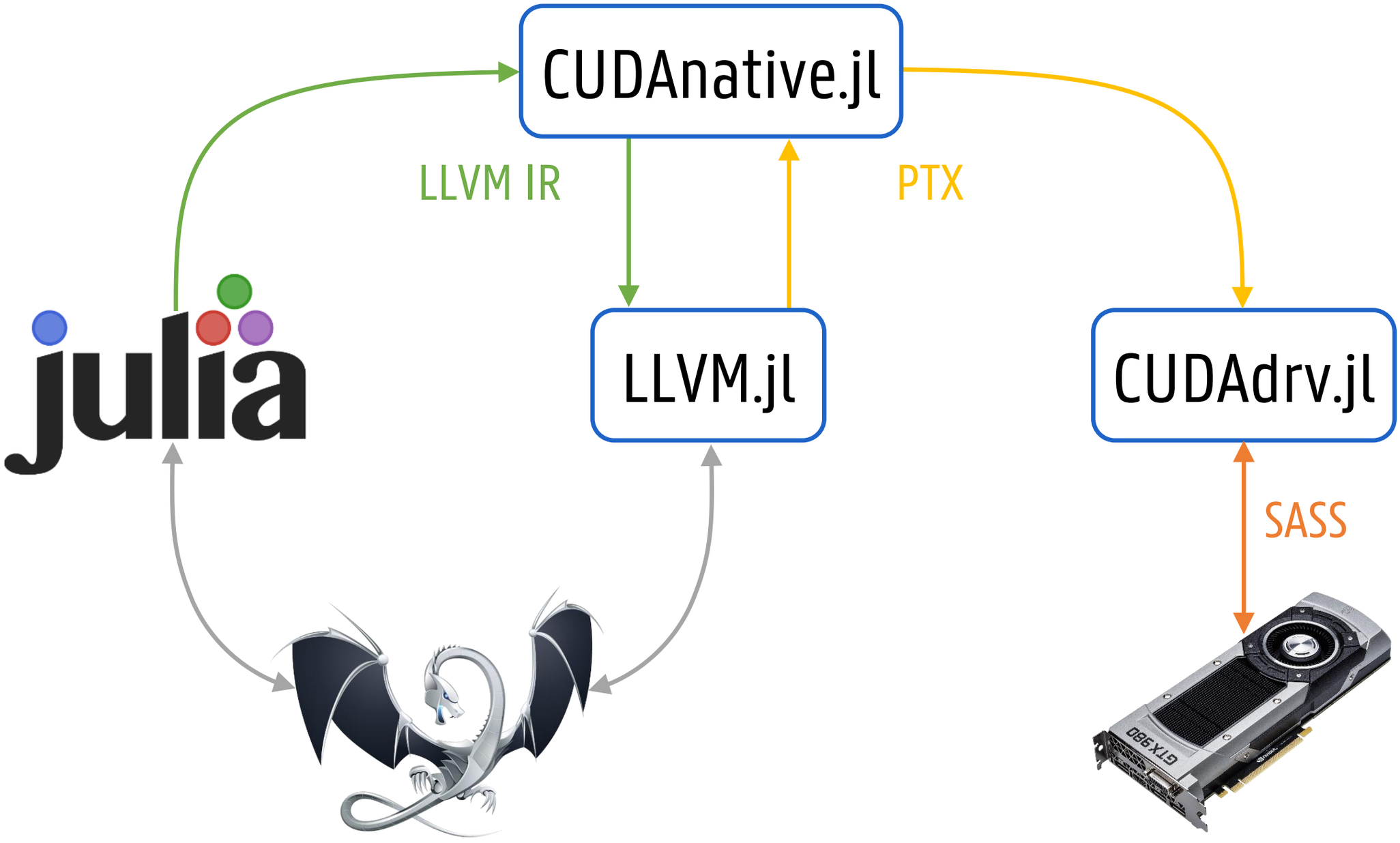
Keep kernels simple!

GHENT UNIVERSITY

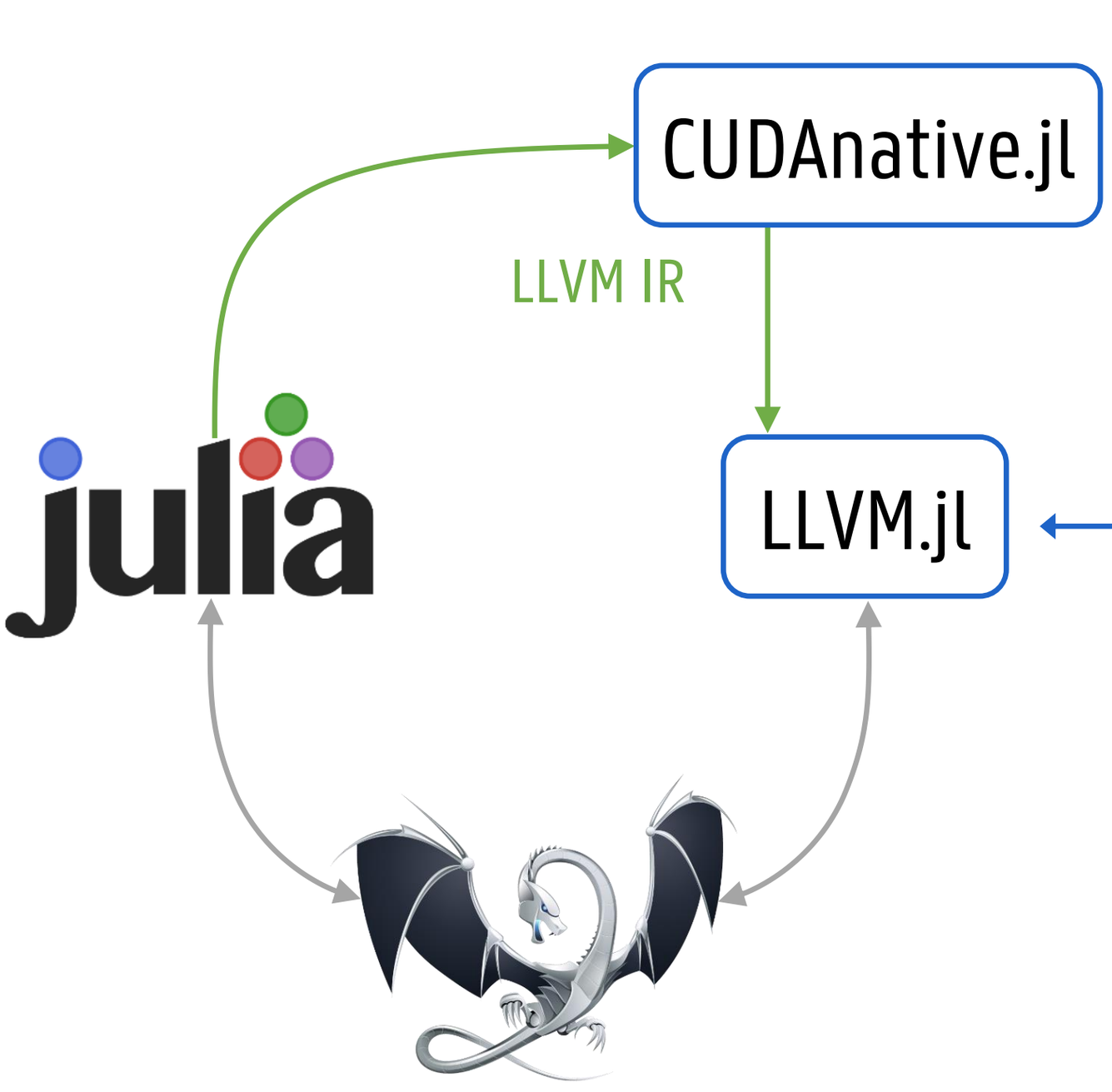
BEHIND THE SCENES

GHENT UNIVERSITY

THE BIG PICTURE



CODE GENERATION




```
InferenceParams
InferenceHooks
CodegenParams
CodegenHooks
```

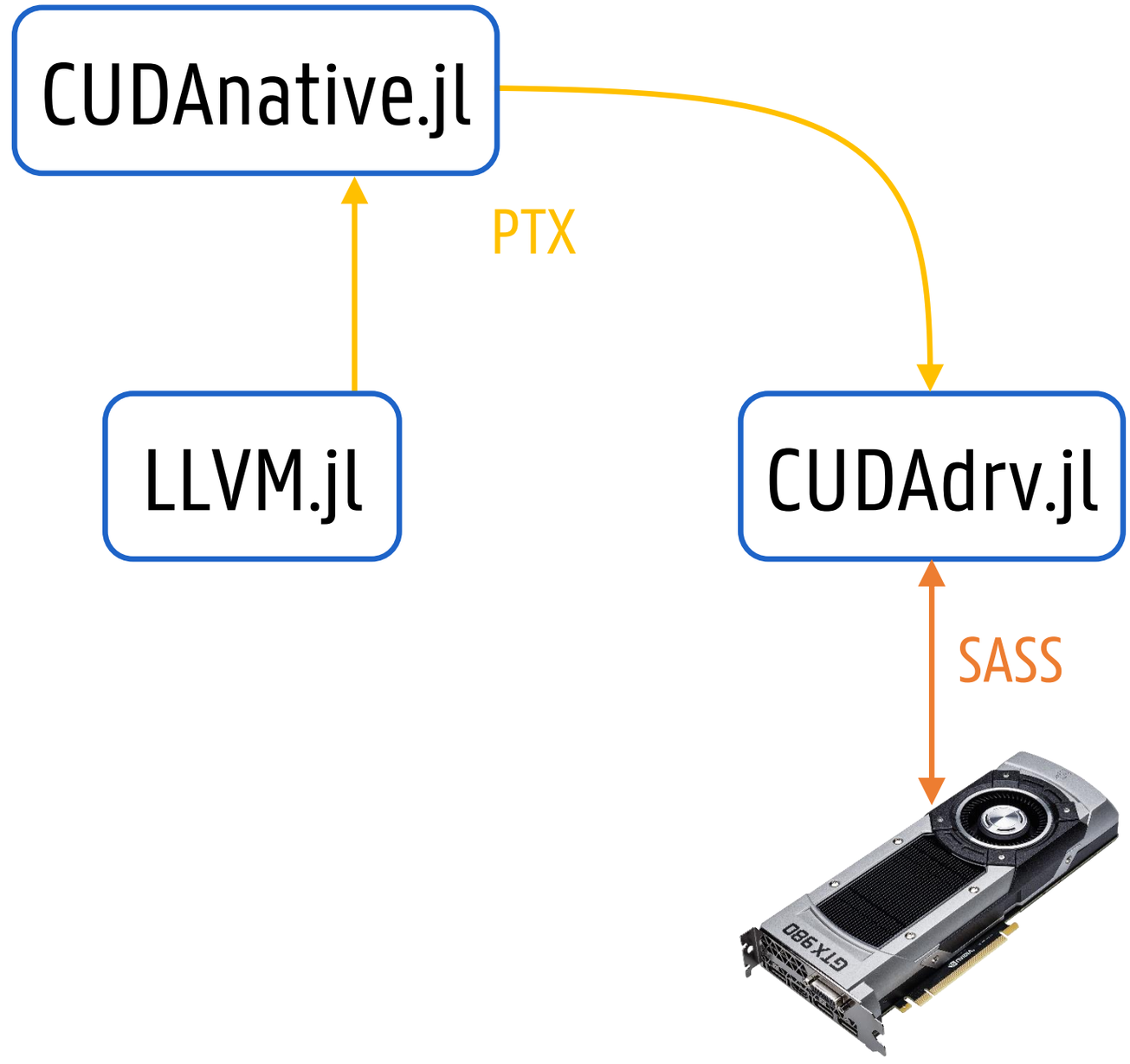
```
emit_exception
```

- link
- optimize
- finalize module

CODE GENERATION

~~NVIDIA: NVVM~~

LLVM: NVPTX



CUDA driver JIT

CODE REFLECTION

```
julia> function add_one(data)
           i = threadIdx().x
           data[i] += one(eltype(data))
           return
       end
```

CODE REFLECTION

```
julia> function add_one(data)

julia> CUDAnative.code_llvm(add_one,
                            Tuple{CuDeviceVector{Int32}})

define void @julia_add_one (%CuDeviceArray*) {
    %1 = tail call i32
        @llvm.nvvm.read.ptx.sreg.ctaid.x()
    ...
    store i32 %9, i32* %7, align 8
    ret void
}
```

CODE REFLECTION

```
julia> function add_one(data)

julia> CUDAnative.code_llvm(add_one,
                            Tuple{CuDeviceVector{Int32}})

julia> a = CuArray{Int32}(N)
julia> CUDAnative.@code_llvm add_one(a)
julia> CUDAnative.@code_llvm @cuda (1,N) add_one(a)
```

GHENT UNIVERSITY

CODE REFLECTION

```
julia> function add_one(data)

julia> @code_ptx add_one(a)

.func add_one(.param .b64 param0) {
    mov.u32 %r2, %ctaid.x;
    ...
    st.u32  [%rd10+-4], %r8;
    ret;
}
```

GHENT UNIVERSITY

CODE REFLECTION

```
julia> function add_one(data)

julia> @code_sass add_one(a)

Function : add_one
    S2R R0, SR_CTAID.X;
    ...
    ST.E [R4], R0;
    EXIT;
```

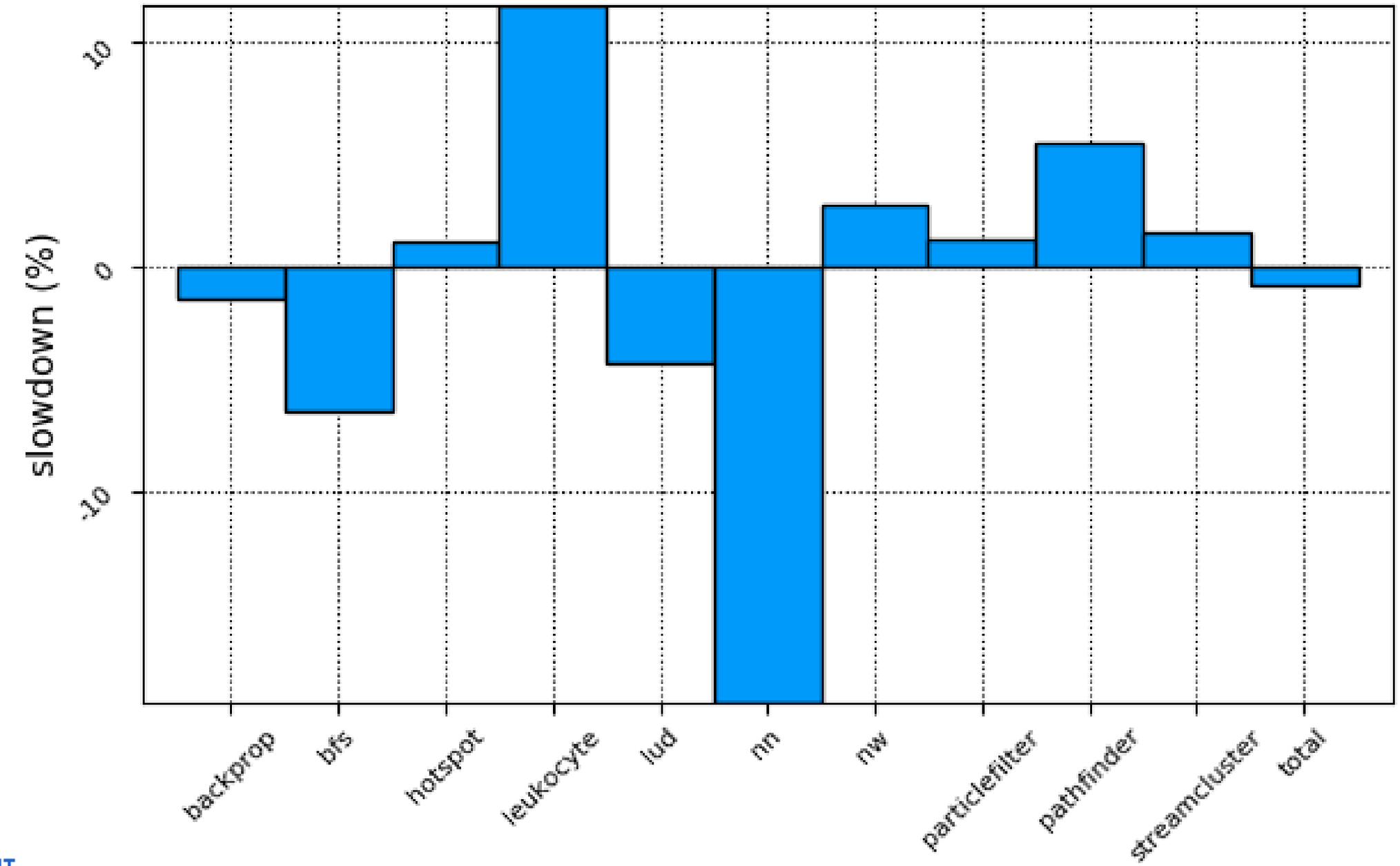
GHENT UNIVERSITY

PERFORMANCE

KERNEL PERFORMANCE

Now 0.77% faster!

https://github.com/JuliaParallel/rodinia



GHENT UNIVERSITY

LAUNCH PERFORMANCE

CUDA CPU: 12.8 µs GPU: 6.8 µs

```
void kernel_dummy(float *ptr)
{
    ptr[0] = 0;
}
```

```
gettimeofday

cuEventRecord
cuLaunchKernel
cuEventRecord

cuEventSynchronize

gettimeofday
```

CUDAdrv.jl CPU: 12.4 µs GPU: 6.9 µs

```
cuModuleLoad

cuModuleGetFunction
```

```
Base.@elapsed begin
    CUDAdrv.@elapsed begin
        cudacall
    end
end
```

GHENT UNIVERSITY

LAUNCH PERFORMANCE

CUDA

CPU: 12.8 µs
GPU: 6.8 µs

```
void kernel_dummy(float *ptr)
{
    ptr[0] = 0;
}
```

```
gettimeofday

cuEventRecord
cuLaunchKernel
cuEventRecord

cuEventSynchronize

gettimeofday
```

CUDAnative.jl

CPU: 12.6 µs
GPU: 7.0 µs

```
function kernel(ptr)
    unsafe_store(ptr, 0f0, 0)
    return
end
```

```
Base.@elapsed begin
    CUDAdrv.@elapsed begin
        @cuda
    end
end
```

GHENT UNIVERSITY

FUTURE WORK

- ❑ Usability
- ❑ Julia support
- ❑ CUDA support
- ❑ Better compiler integration

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

DEPARTMENT ELECTRONICS AND INFORMATION SYSTEMS
COMPUTER SYSTEMS LAB

PROGRAMMING NVIDIA GPUS WITH CUDANATIVE.JL

https://github.com/JuliaGPU/CUDAnative.jl

https://github.com/JuliaGPU/CUDAdrv.jl

https://github.com/maleadt/LLVM.jl

Tim Besard – 2017-06-21

GHENT UNIVERSITY