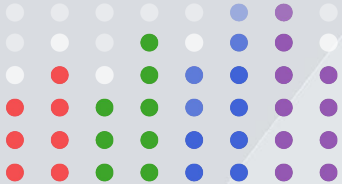


CUDA.jl

Update on new features and developments

Julia computing

Tim Besard

www.juliacomputing.com

What has changed?

Since JuliaCon'21 (v3.3.4)

- 244 files changed, 11978 insertions(+), 6383 deletions(-) (excluding generated headers, Artifacts, …)
- 45 contributors (25 new)
- 685 commits

… so quite a lot!

10 minute overview: array programming, kernel programming

Unified memory (v3.4)

CuArray now keeps track of buffer type:

```
julia> a = CUDA.zeros(1)
1-element CuArray{Float32, 1, CUDA.Mem.DeviceBuffer}:
 0.0

julia> b = CuVector{Float32, Mem.UnifiedBuffer}(undef, 1)
1-element CuArray{Float32, 1, CUDA.Mem.UnifiedBuffer}:
 0.0

julia> copyto!(b, [1])
1-element CuArray{Float32, 1, CUDA.Mem.UnifiedBuffer}:
 1.0

julia> c = cu([0]; unified=true)
1-element CuArray{Int64, 1, CUDA.Mem.UnifiedBuffer}:
 0
```

```
julia> @benchmark CUDA.@sync copyto!(a, [1])
 Range (min … max):  8.420 μs …  32.049 μs
 Time  (median):     9.290 μs
 Time  (mean ± σ):   9.365 μs ± 380.323 ns

julia> @benchmark CUDA.@sync copyto!(b, [1])
 Range (min … max):  826.526 ns …  28.862 μs
 Time  (median):     855.513 ns
 Time  (mean ± σ):   865.160 ns ± 388.405 ns
```

Copies between devices (v3.8)

```
julia> device!(0);
julia> a = CUDA.rand(2,2)
2×2 CuArray{Float32, 2, CUDA.Mem.DeviceBuffer}:
 0.440147  0.986939
 0.622901  0.698119

julia> device!(1);
julia> b = CUDA.zeros(2,2);

julia> copyto!(b, a)
2×2 CuArray{Float32, 2, CUDA.Mem.DeviceBuffer}:
 0.440147  0.986939
 0.622901  0.698119
```

Automatically:

- uses NVLINK
- (or) uses PCIe
- (or) stages through CPU

`isbits` union support (3.3)

```
julia> a = CuArray([1, nothing, 3])
3-element CuArray{Union{Nothing, Int64}, 1}:
 1
  nothing
 3

julia> findfirst(isnothing, a)
2
```

Sparse array improvements

limit csc/csr/bsr sparse conversion index to be cint & fix a few conversion bugs ✗ cuda array
#1563 opened 7 days ago by Roger-luo • Draft

specialize +/- op for sparse diag ✓ 1 8
#1514 by Roger-luo was merged on 14 Jun

CUSPARSE: Support mixed type mv ✓ cuda libraries enhancement 12
#1475 by Roger-luo was merged on 18 May

support CuSparseMatrix(::Diagonal) ✓ cuda array enhancement 2
#1470 by Roger-luo was merged on 11 Apr

CUSPARSE: Better error msg for unsupported sparse mm ✗ cuda libraries 5
#1467 opened on 7 Apr by Roger-luo

support sparse opnorm ✓ cuda libraries 9
#1466 by Roger-luo was merged on 9 May

Sparse array broadcast (3.9)

```
julia> cx = sprand(Float32, 1024, 1024, 0.1);
julia> cy = sprand(Float32, size(cx)..., 0.1);
julia> @benchmark cx .+ cy
 Range (min … max):  977.180 μs …   3.316 ms
 Time  (median):     984.579 μs
 Time  (mean ± σ):     1.016 ms ± 111.271 μs
```

```
julia> x = CuSparseMatrixCSR(cx);
julia> y = CuSparseMatrixCSR(cy);
julia> @benchmark CUDA.@sync x .+ y
 Range (min … max):  158.108 μs …  12.491 ms
 Time  (median):     166.548 μs
 Time  (mean ± σ):   170.143 μs ± 211.001 μs
```

```
julia> x = CuSparseMatrixCSC(cx);
julia> y = CuSparseMatrixCSC(cy);
julia> @benchmark CUDA.@sync x .+ y
 Range (min … max):  154.798 μs …  12.787 ms
 Time  (median):     159.659 μs
 Time  (mean ± σ):   163.956 μs ± 217.046 μs
```

Output remains sparse!

Device capability-dependent code (v3.4)

```
function kernel(a)
    a[] = compute_capability() >= sv"6.0" ? 1 : 2
    return
end
```

```
julia> CUDA.code_llvm(kernel, Tuple{CuDeviceVector{Float32, AS.Global}})
define void @julia_kernel_1({ i8 addrspace(1)*, i64, [1 x i64] }* %0) {
top:
  %1 = bitcast { i8 addrspace(1)*, i64, [1 x i64] }* %0 to float addrspace(1)**
  %2 = load float addrspace(1)*, float addrspace(1)** %1, align 8
  store float 1.000000e+00, float addrspace(1)* %2, align 4
  ret void
```

Device capability-dependent code (v3.4)

```
function kernel(a)
    a[] = compute_capability() >= sv"6.0" ? 1 : 2
    return
end
```

```
julia> capability(device!(1))
v"3.5.0"

julia> CUDA.code_llvm(kernel, Tuple{CuDeviceVector{Float32, AS.Global}})
define void @julia_kernel_2({ i8 addrspace(1)*, i64, [1 x i64] }* %0) {
top:
  %1 = bitcast { i8 addrspace(1)*, i64, [1 x i64] }* %0 to float addrspace(1)**
  %2 = load float addrspace(1)*, float addrspace(1)** %1, align 8
  store float 2.000000e+00, float addrspace(1)* %2, align 4
  ret void
}
```

Supported functions: `compute_capability()` and `ptx_isa_version()`

Improved atomic operations (3.4)

Low-level API

```
julia> function kernel(a)
         CUDA.atomic_add!(pointer(a), one(eltype(a)))
         return
       end

julia> a = CUDA.zeros(1);
julia> @cuda kernel(a);
julia> a
1-element CuArray{Float32, 1, CUDA.Mem.DeviceBuffer}:
 1.0
```

Improved atomic operations (3.4)

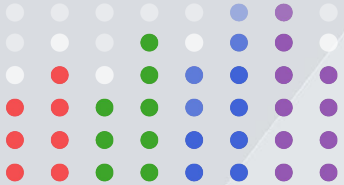
High-level API

```
julia> function kernel(a)
         CUDA.@atomic a[] += one(eltype(a))
         return
       end

julia> @cuda kernel(a);
julia> a
1-element CuArray{Float32, 1, CUDA.Mem.DeviceBuffer}:
 2.0
```

Forward compatibility (3.5)

```
julia> CUDA.versioninfo()
CUDA toolkit 11.7, artifact installation
NVIDIA driver 510.47.3, for CUDA 11.6
CUDA driver 11.7
```



CUDA.jl

https://juliagpu.org/

Julia computing

Tim Besard