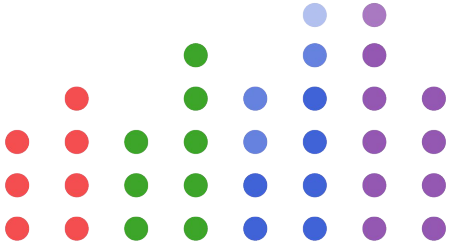


Julia - A fresh approach to computing

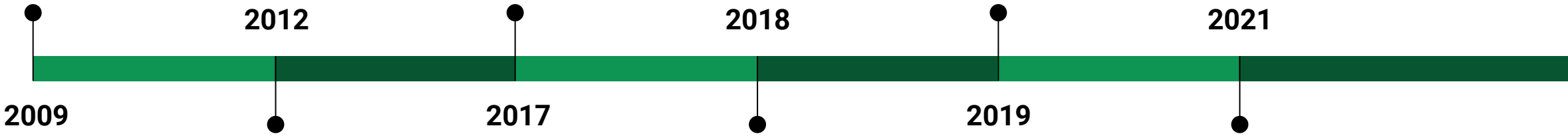
Dr. Viral B. Shah

Co-creator, Julia
Co-founder and CEO, Julia Computing

Julia computing

julia

Julia has been an 11 year journey so far...



An urge to solve the two language problem

Why we created Julia

In 2013, Julia becomes the "Ju" in Jupyter

1M Julia downloads

Releases:

1. DifferentialEquations.jl
2. Flux.jl

Julia 1.0 released

10M Julia downloads
Wilkinson Prize

30M Julia downloads
Julia 1.7 released

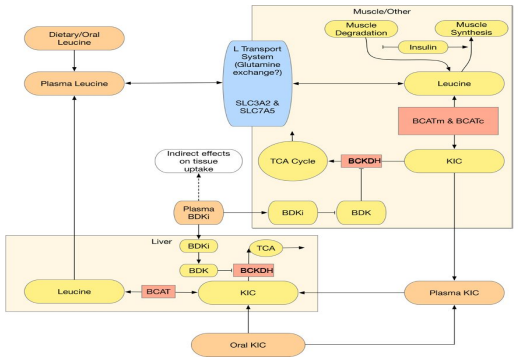
Julia computing

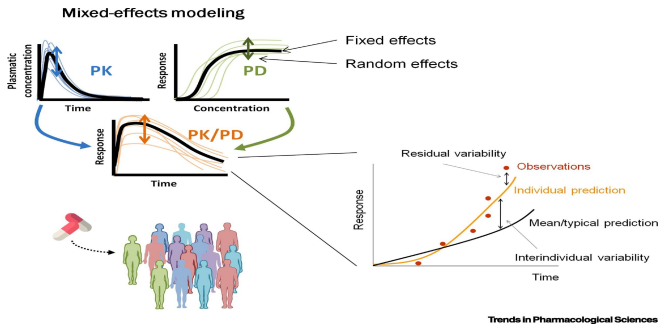
JuliaHub

August 2021

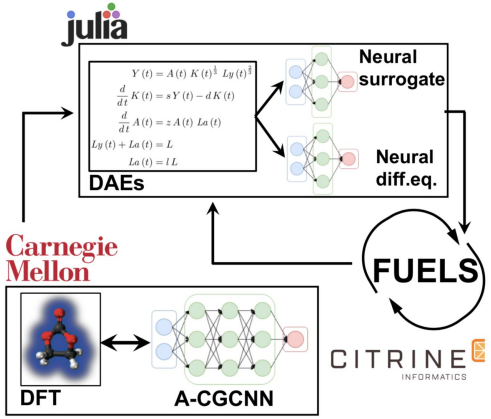
… And it is making real impact

Accelerating Drug Development (175x)





Designing Efficient batteries (10x)

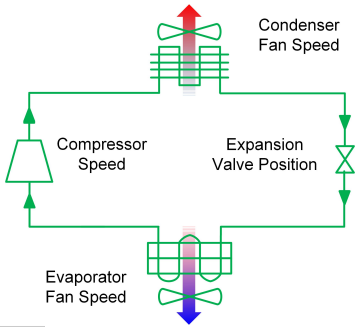


Ford

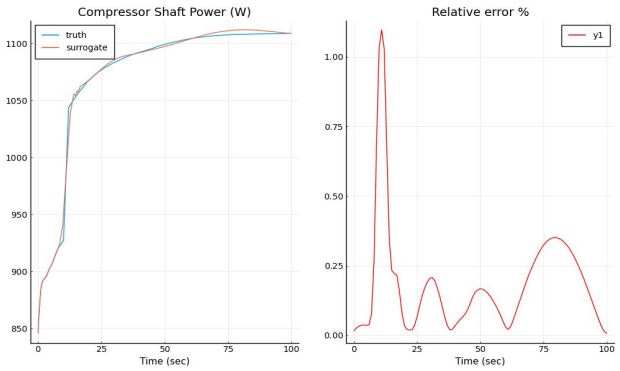
QuantumScape

Modeling Climate, Energy Efficiency (570x)

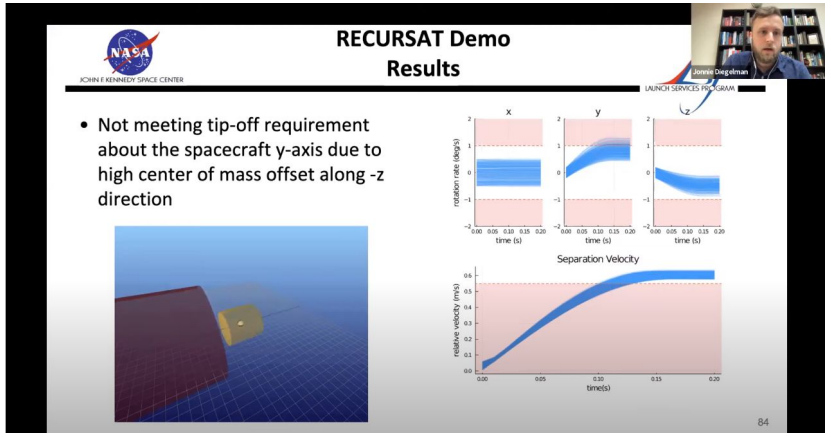
CliMA
CLIMATE MODELING ALLIANCE



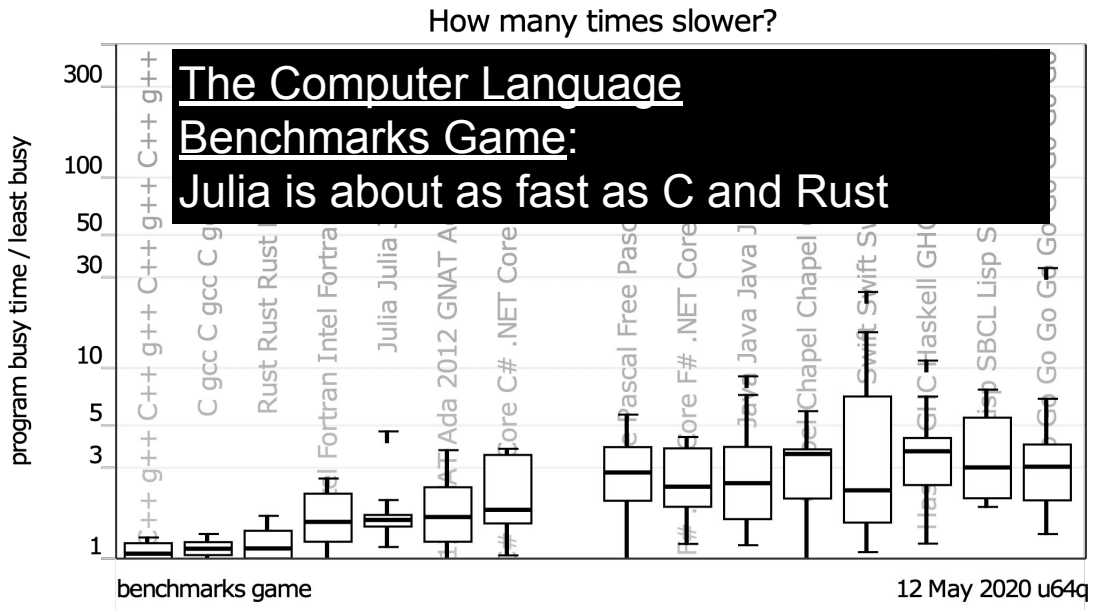
MITSUBISHI ELECTRIC
Changes for the Better
MITSUBISHI ELECTRIC RESEARCH LABORATORIES, INC



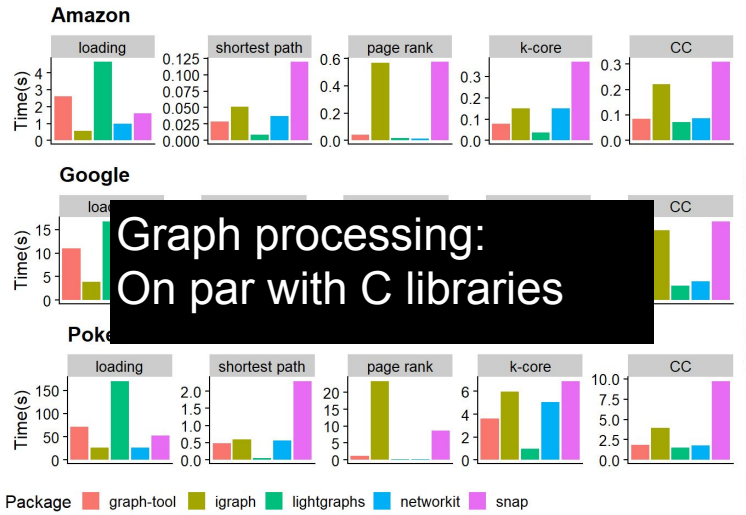
Space mission planning (15,000x)



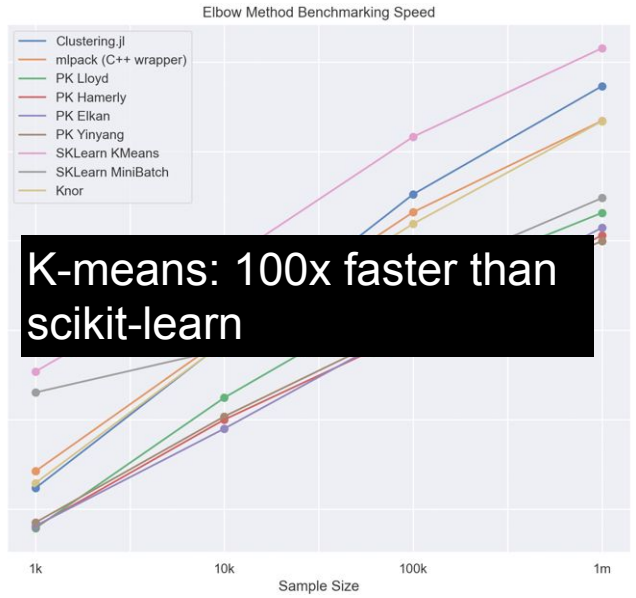
Winning various benchmarks



The Computer Language Benchmarks Game:
Julia is about as fast as C and Rust



Graph processing:
On par with C libraries

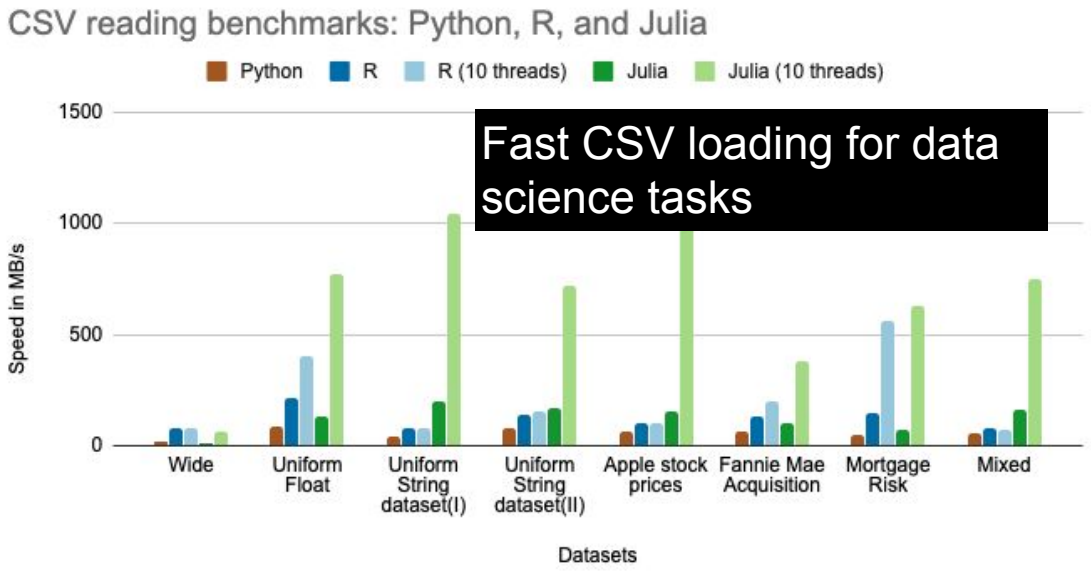


K-means: 100x faster than scikit-learn

Input table: 1,000,000,000 rows x 9 columns (50 GB)

data.table	1.14.1	2021-05-19	149s
DataFrames.jl	1.1.1	2021-05-15	200s
ClickHouse	21.3.2.5	2021-05-12	256s
cuDF*	0.19.2	2021-05-31	492s
spark	3.1.1	2021-03-09	537s
(py)datatable	1.0.0a0	2021-05-19	756s
dplyr	1.0.6	2021-05-08	internal error
pandas	1.2.4	2021-04-29	out of memory
dask	2021.04.1	2021-05-09	out of memory
Polars	0.7.18	2021-05-24	out of memory
Arrow			internal error
Duck			memory
Modin			pending

H2O DataFrames benchmark:
Scales to large datasets



Fast CSV loading for data science tasks

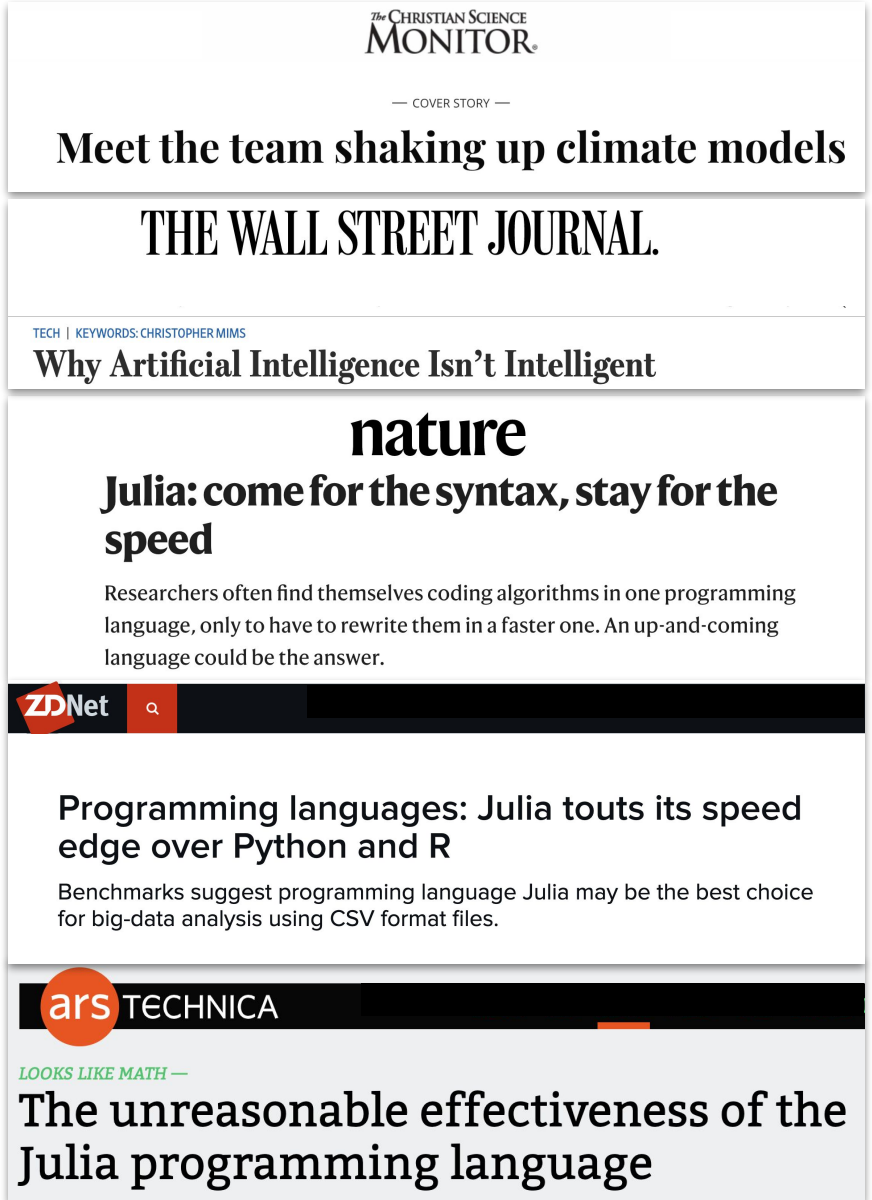
The Julia Community

Julia has entered the mainstream

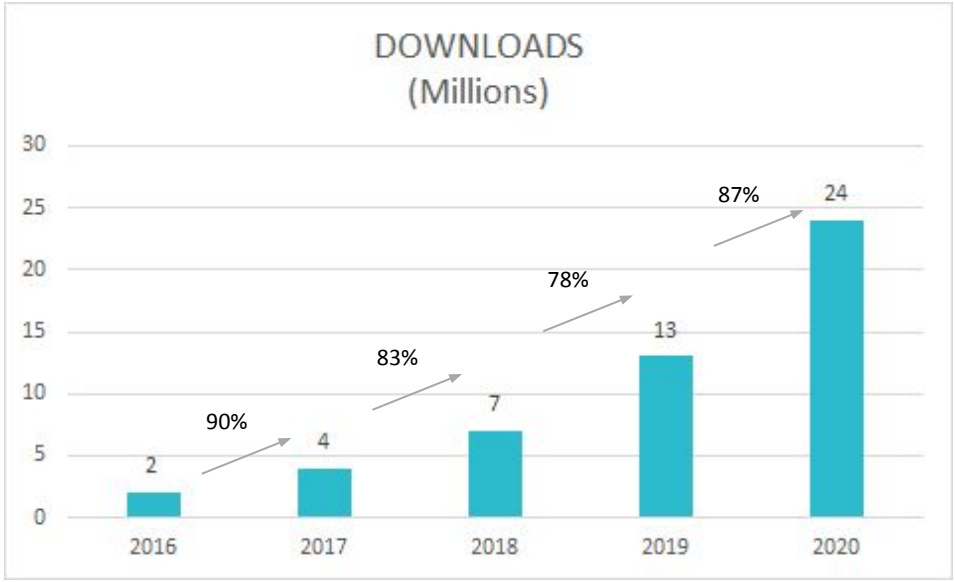
- **#20** on IEEE Spectrum
- **#27** on Tiobe Index (from #47 in 2020)
- **#22** on PYPL (PopularitY of Programming Language Index)

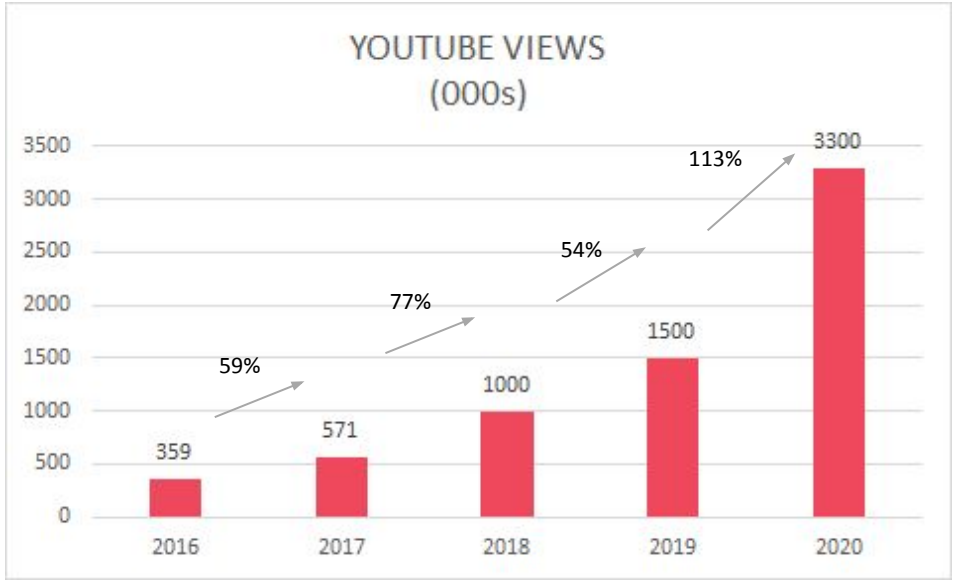
Community

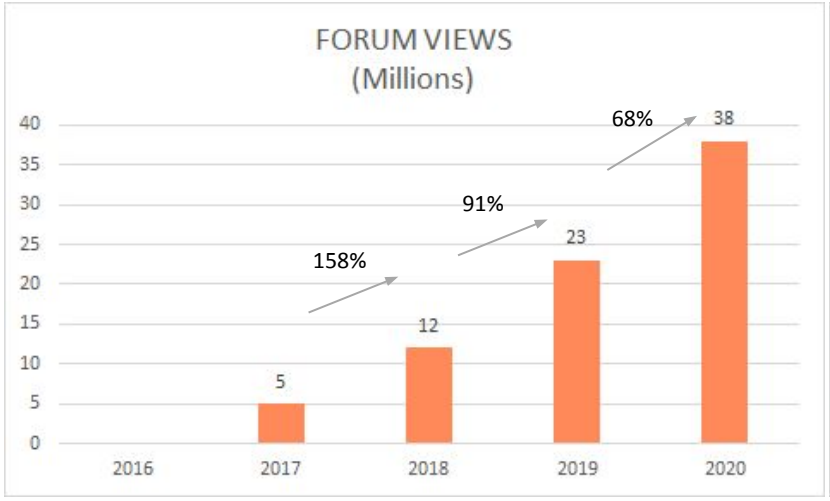
- Users: 1M
- The 8th Annual JuliaCon was held this year (43,000 visitors)
- Downloaded by over 10,000 companies worldwide
- Downloaded by over 1,500 universities
  - Taught at MIT, Stanford, Berkeley, and many others

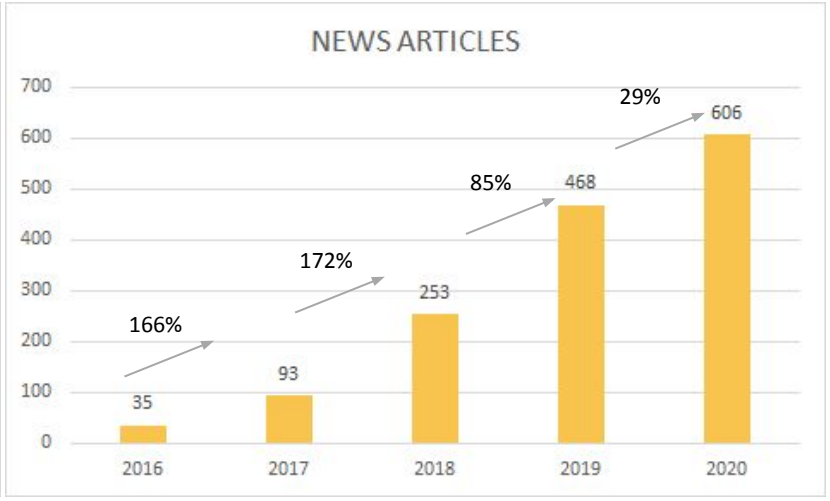


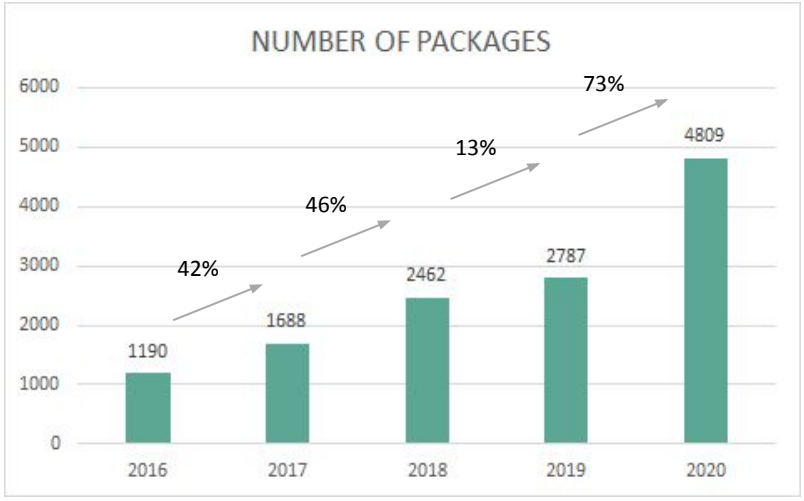
The Julia community continues to grow exponentially





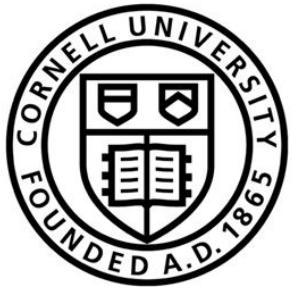


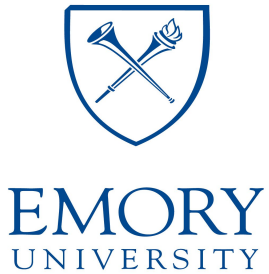


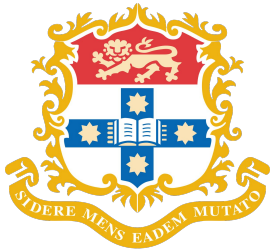


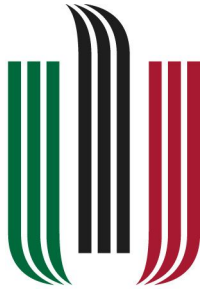
All numbers are cumulative

Over 1,500 universities are using and teaching Julia









The 8th annual JuliaCon will be held online this year

JuliaCon 2020 had 30,000 online visitors, JuliaCon 2021 had 43,000 online visitors!

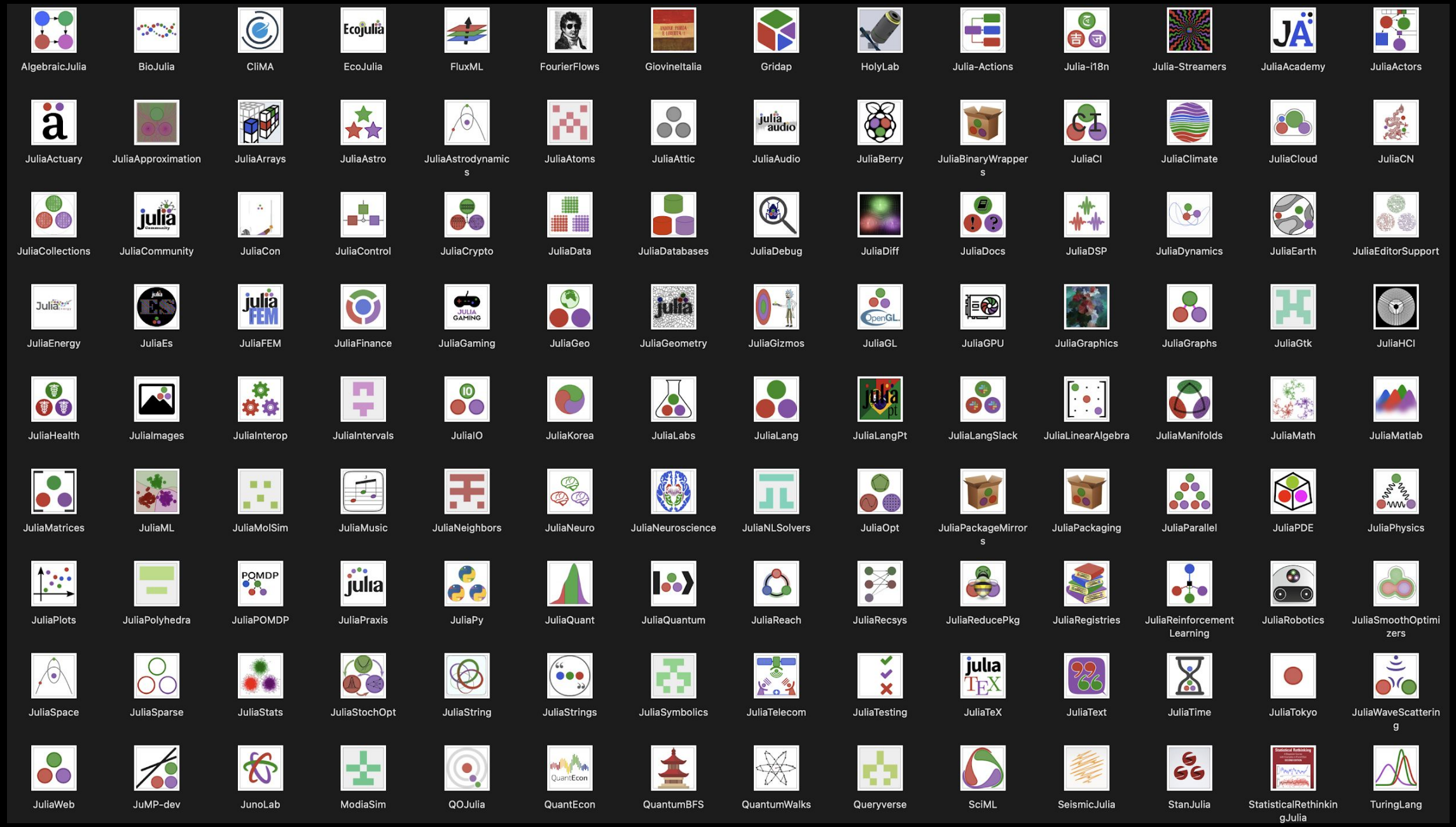

MIT 2015, 2016

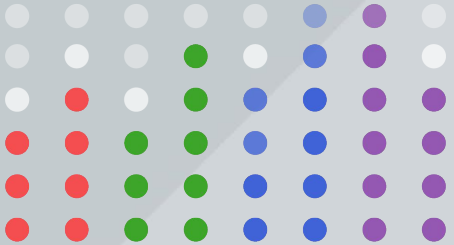

London 2018


Baltimore 2019


Berkeley 2017

Composing across 100+ sub-communities





Composing with Multiple Dispatch

Language for describing what to specialize on.

- Design descriptive types for the domain at hand
- Write methods for whatever cases you can handle
- Compiler generates specializations
- These need to be *de-coupled*

JuliaCon 2017: The Unreasonable Effectiveness of Multiple Dispatch

Sliding scale of specialization

`Array`	Some kind of array
`Array{Int}`	Element type known to be Int
`Array{Int,2}`	… and 2-dimensional
`Array{<:Any,2}`	… or unknown element type
`Array{<:Real,2}`	… or unknown, but `Real`, element type
`SArray{(2,3),Float64,2}`	2d `Float64` with dimensions 2x3

Example

```
1 + 2.0
```

⇓

```
Tuple{typeof(+), Int64, Float64}
```

```
# as a method definition
+(x::Int64, y::Float64) = …

julia> methods(+)
# 208 methods for generic function "+":
```

Example: one-hot vector type

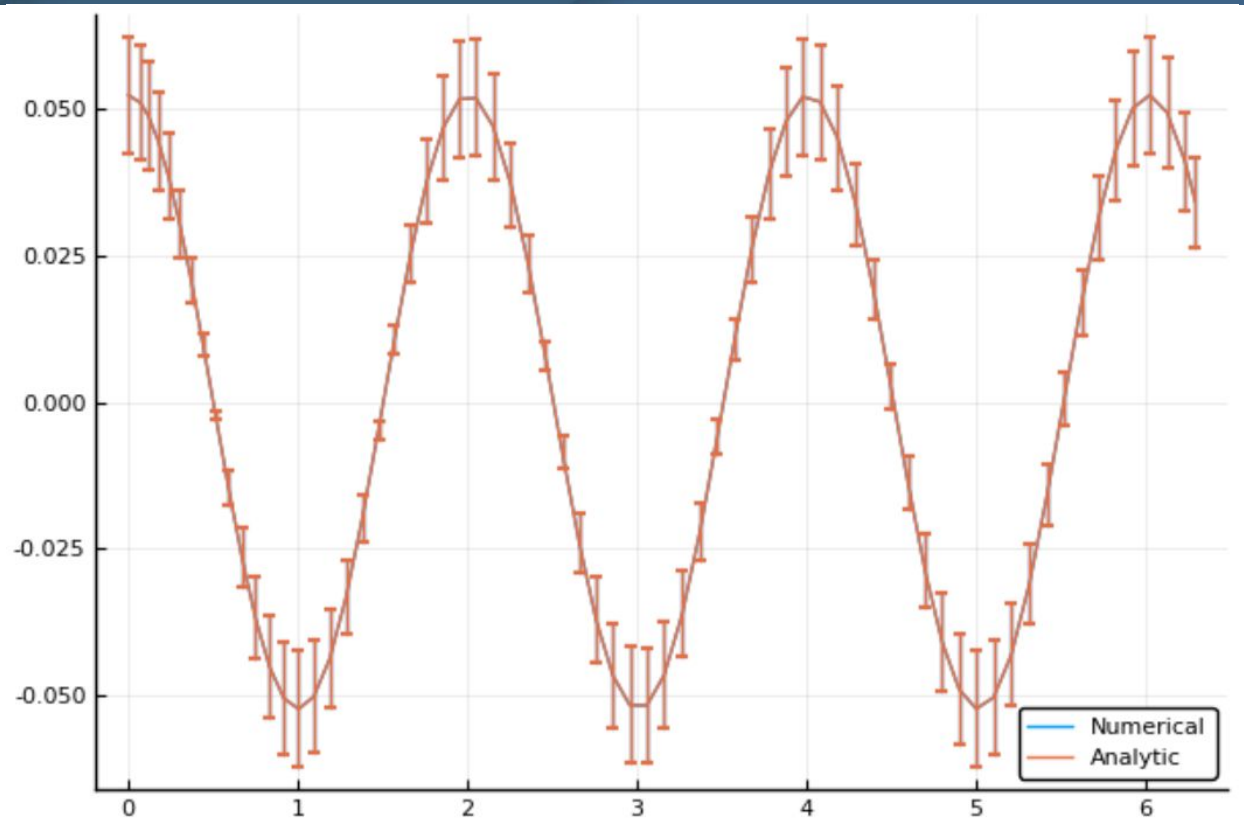
```
struct OneHotVector <: AbstractVector{Bool}
    index::Int
    len::Int
end

size(xs::OneHotVector)                    = (xs.len,)

getindex(xs::OneHotVector, i::Integer) = i == xs.index

A::AbstractMatrix * b::OneHotVector    = A[:, b.index]
```

Composability: Other special matrix types

- `Diagonal`
- `UniformScaling`
- `Symmetric, Hermitian`
- `LowerTriangular, UpperTriangular`
- `Bidiagonal, Tridiagonal, SymTridiagonal`
- `Adjoint, Transpose`

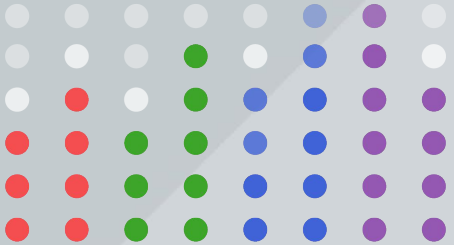
Composability: DifferentialEquations.jl + Measurements.jl

```julia
g = 9.79 ± 0.02 # Gravitational constants
L = 1.00 ± 0.01 # Length of the pendulum

# Initial speed & angle, time span
u₀ = [0 ± 0, π/60 ± 0.01]
tspan = (0.0, 6.3)

# Define the problem
function pendulum(du, u, p, t)
    θ  = u[1]
    dθ = u[2]
    du[1] = dθ
    du[2] = -(g/L)*θ
end

# Pass to solvers
prob = ODEProblem(pendulum, u₀, tspan)
sol = solve(prob, Tsit5(), reltol = 1e-6)

# Analytic solution
u = u₀[2] .* cos.(sqrt(g/L) .* sol.t)
```



Rackauckas et al. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. 2017. (*Journal of Open Research Software*)

Giordano. Uncertainty propagation with functionally correlated quantities (*arXiv:1610.08716*)

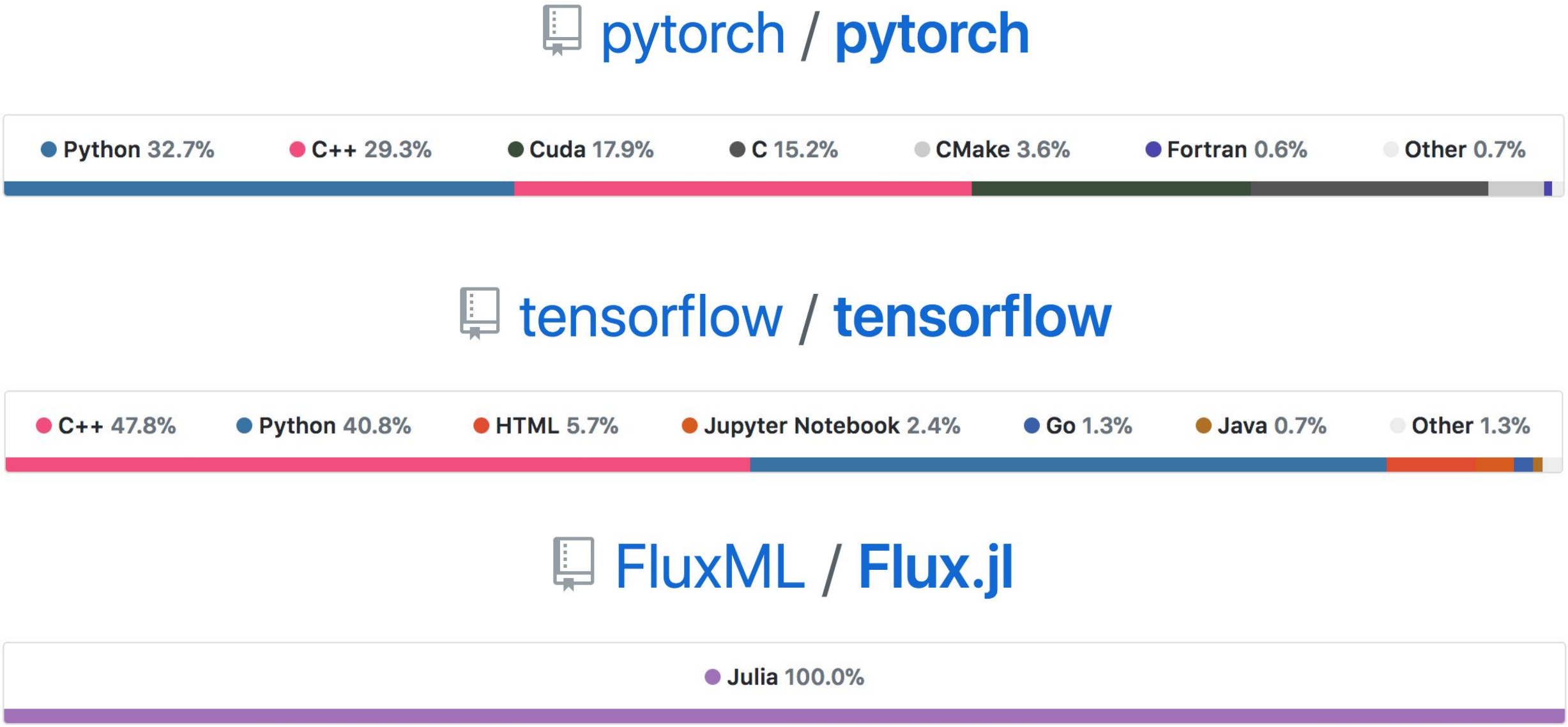


Key Idea

Models are really programs

ML problems have benefited from language infrastructure

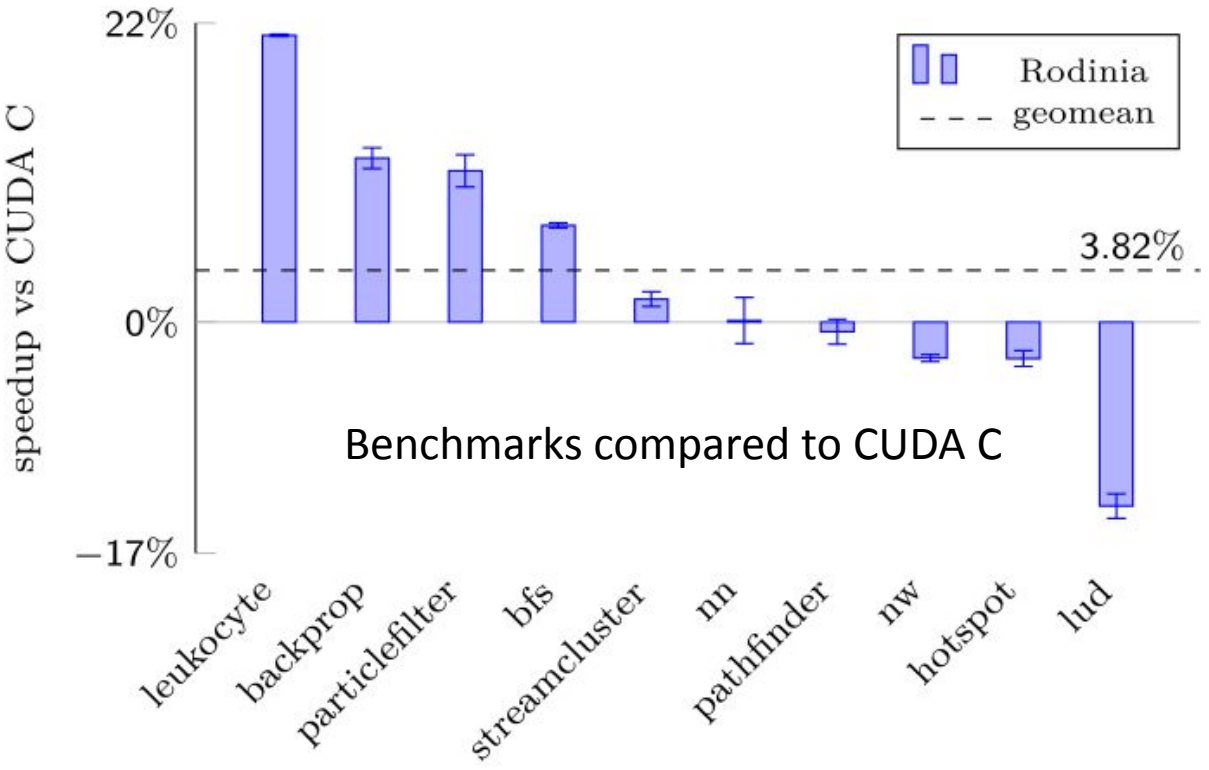
Modeling and Simulation problems can also
benefit from language infrastructure



Fashionable Modelling with Flux
(arXiv:1811.01457)

Julia on GPUs: https://juliagpu.org

Supports NVIDIA GPUs. Early support for Intel and AMD GPUs.



Noteworthy new capabilities

- Multi-GPU programming
- Support for CUDA 11
- Multi-tasking and multi-threading

Noteworthy applications

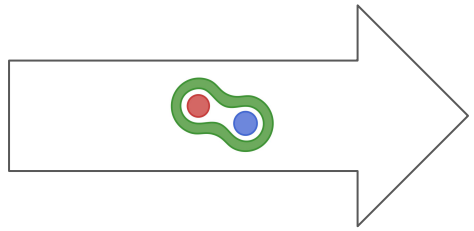
- 175x improvement in pharmaceutical workloads
- 1,000 GPU parallel deployment at CSCS (Switzerland)
- Clima Project – Oceananigans.jl
- Multi-physics simulations
- Reinforcement learning – AlphaZero.jl

Tim Besard et al. Effective extensible programming: Unleashing Julia on GPUs. IEEE Transactions on Parallel and Distributed Systems (2018).

Differentiable Programming in Julia

Zygote.jl, Diffractor.jl, ForwardDiff.jl, ChainRules.jl

```
function foo(W, Y, x)
    Z = W * Y
    a = Z * x
    b = Y * x
    c = tanh.(b)
    r = a + c
    return r
end
```



```
function ∇foo(W, Y, x)
    Z = W * Y
    a = Z * x
    b = Y * x
    c, 𝒥thanh = ∇tanh.(b)
    a + c, function (Δr)
        Δc = Δr, Δa = Δr
        (Δtanh, Δb) = 𝒥thanh(Δc)
        (ΔY, Δx) = (Δb * x', Y' * Δb)
        (ΔZ = Δa * x', Δx += Z' * Δa)
        (ΔW = ΔZ * Y', ΔY = W * ΔZ')
        (nothing, ΔW, ΔY, Δx)
    end
end
```

Don't Unroll Adjoint: Differentiating SSA-Form Programs (Systems for ML Workshop, NIPS 2018)

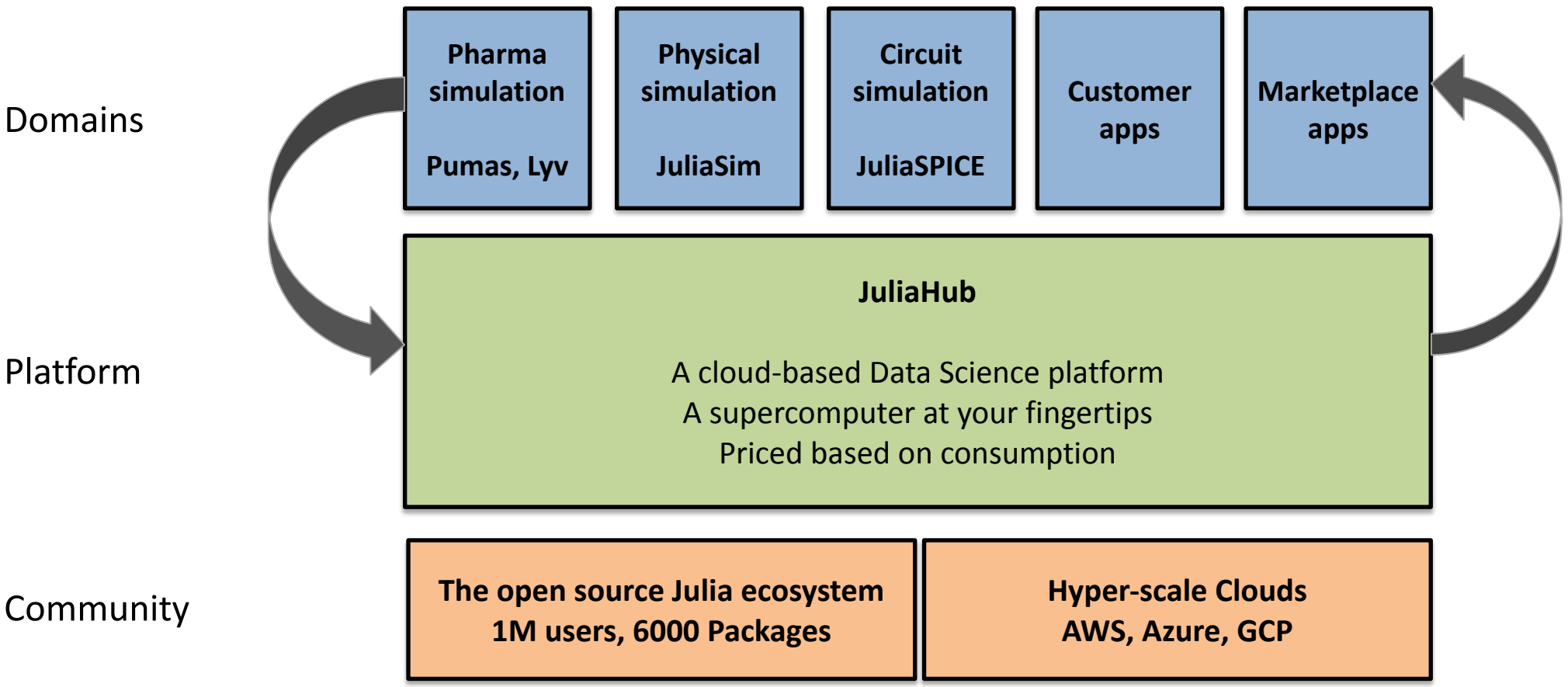
Generalized Physics-Informed Learning through Language-Wide Differentiable Programming. *AAAI Spring Symposium: MLPS 2020*

Composability of compiler transformations

Diffractor.jl, ModelingToolkit.jl

Transforms / Programming Models	Optimizations	Code Generators
Automatic Differentiation SPMD Task Parallelism Data Query Interval Constraint Programming Disciplined Convex Programming Tracing/Profiling/Debugging Verification/Formal Methods . . .	Scalar Tensor Parallel/Distributed Data Layout Data Access (Polyhedral) Relational (Query Compilers) Symbolic . . .	CPU GPU TPU / ML Accelerators Virtual Machines (WebAssembly) FPGA / Silicon Quantum Computers Homomorphic Encryption . . .

A strong business foundation
Community + Platform + Domains



Julia computing

JuliaHub

August 2021