

# Introduction to Julia for High-Performance Computing

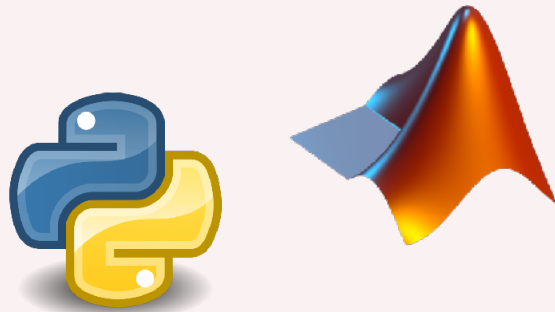
Carsten Bauer @ HLRS, Stuttgart

September 10, 2024

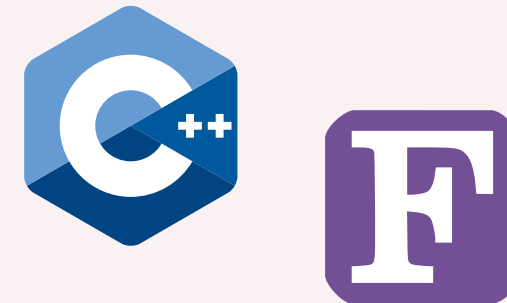
# We don't always speak the same language



Domain Science



High-Performance Computing



Language Barrier

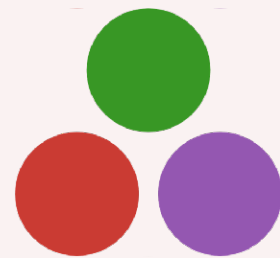
# Julia aims to solve the "two-language problem"



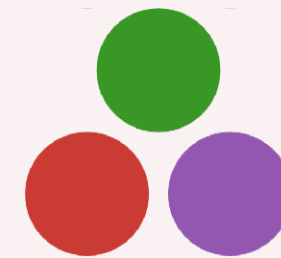
Domain Science



High-Performance Computing



Gradual transition



"Julia: come for the syntax,  
stay for the speed"

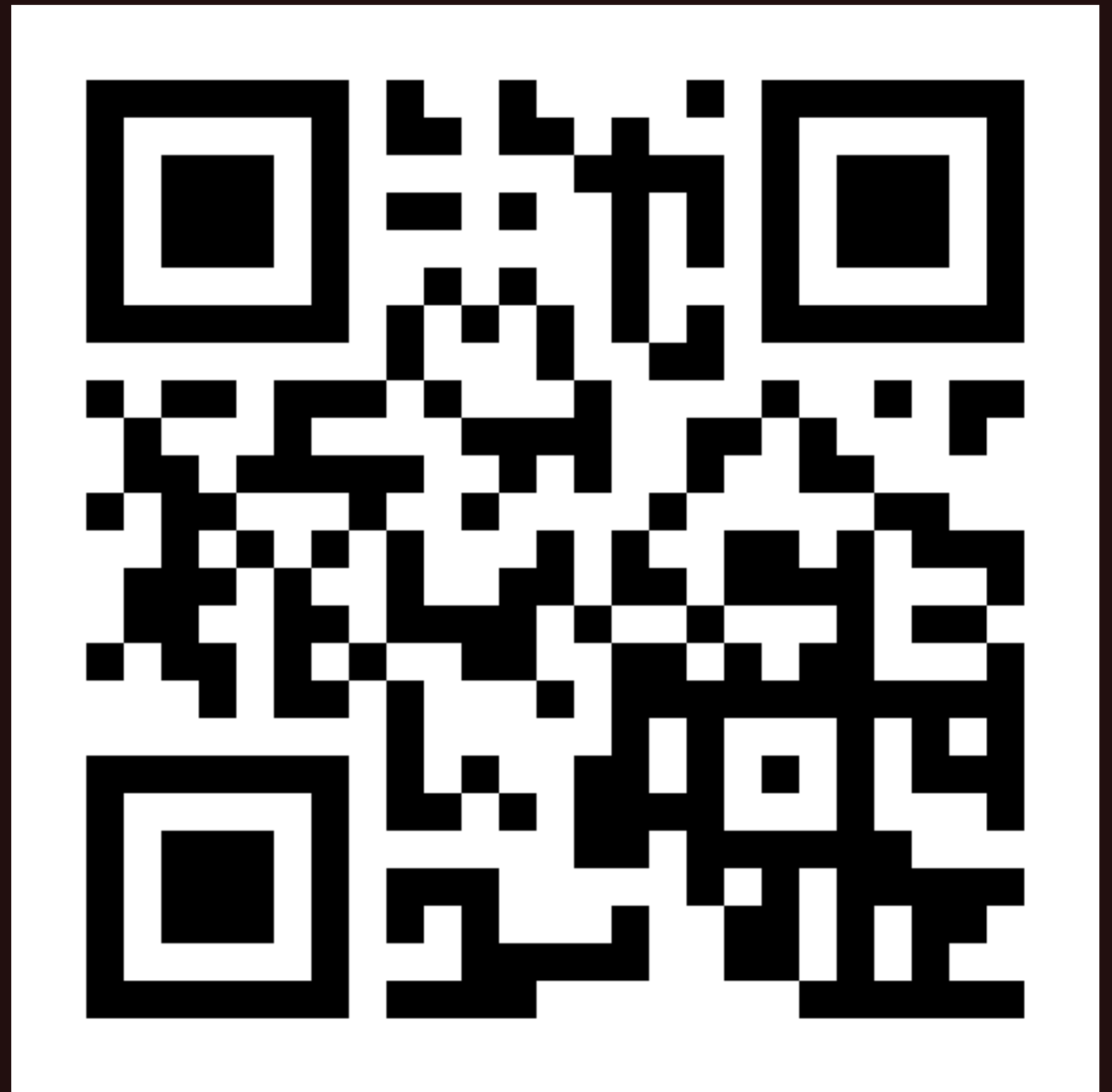
**nature**

# Learn about Julia (for HPC) and HPC (with Julia)

	Tuesday	Wednesday	Thursday	Friday
	Foundations	Core	Node	Cluster
09:00 - 10:45	<div>Intro</div> <div>Onboarding</div>	Type & Memory Optimizations	Multithreading	Distributed Computing
10:45 - 11:00	Break	Break	Break	Break
11:00 - 12:30	Fundamentals	Exercises	Exercises	Exercises
12:30 - 14:00	Lunch	Lunch	Lunch	Lunch
14:00 - 15:30	Specialisation & Abstraction	SIMD & Profiling	GPU Computing	Exercises
15:30 - 15:45	Break	Break	Break	Outro
15:45 - 17:00	Exercises	Exercises	Exercises	

# Quick Survey

<https://etc.ch/wZaG>



# Julia's Weaknesses

HPC with Julia is  
currently a **niche**.

**Limited support** by  
vendors and HPC centers

**Few people** maintain  
many core packages

Still **maturing**

Achieving  
high performance  
can be tricky.

Garbage collection

Type instabilities

Task-based multithreading



No easy way to  
produce (small)  
**shared libraries.**

**PackageCompiler.jl** is currently  
your best bet

Hampers integration into  
existing code bases

# Julia's Strengths

Julia is **interactive**  
and **convenient**.

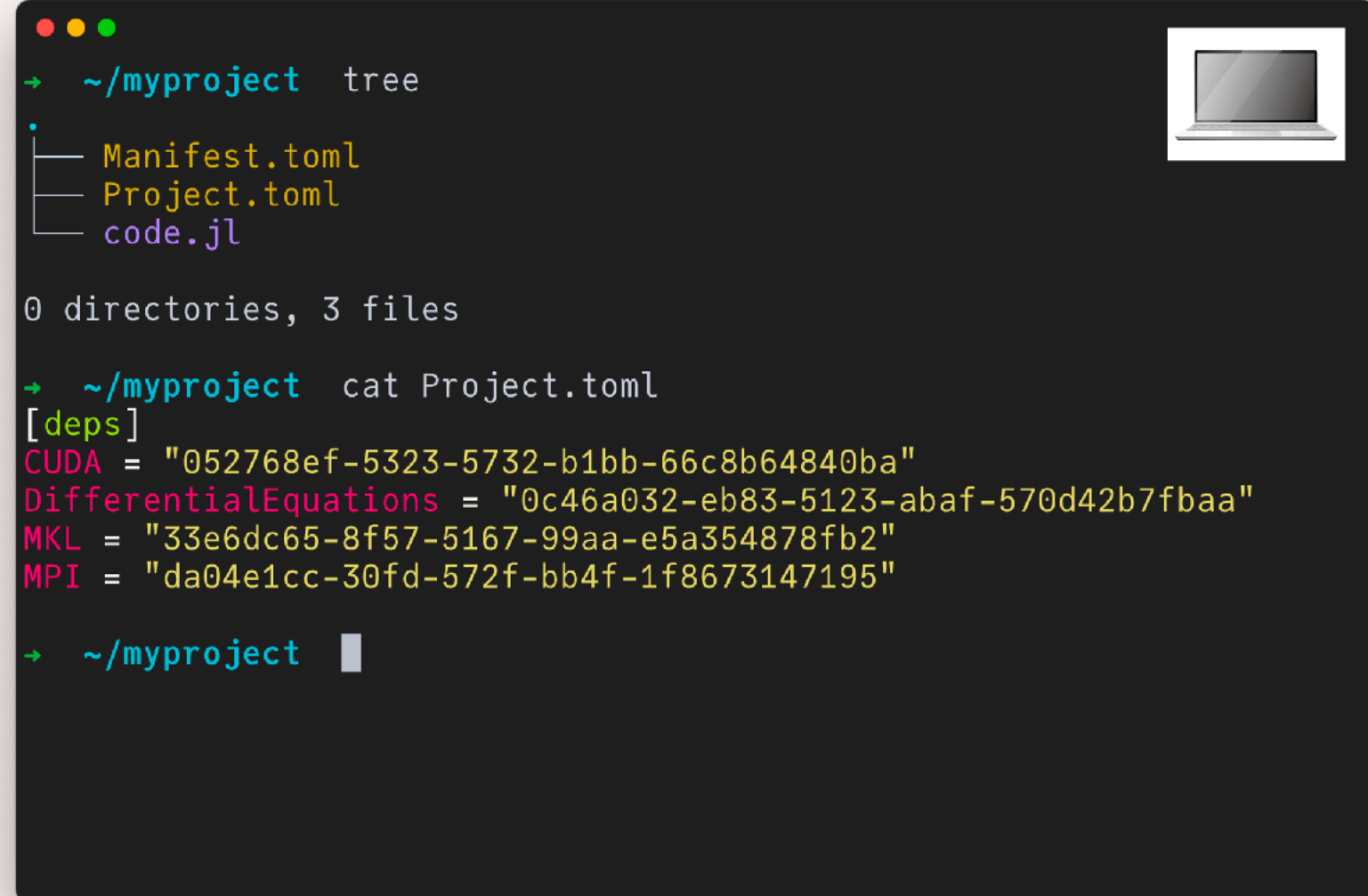
Powerful **REPL**, **Jupyter**, ...

Great **math** support

Best-in-class **package manager**

# Software portability is as good as it gets

Laptop



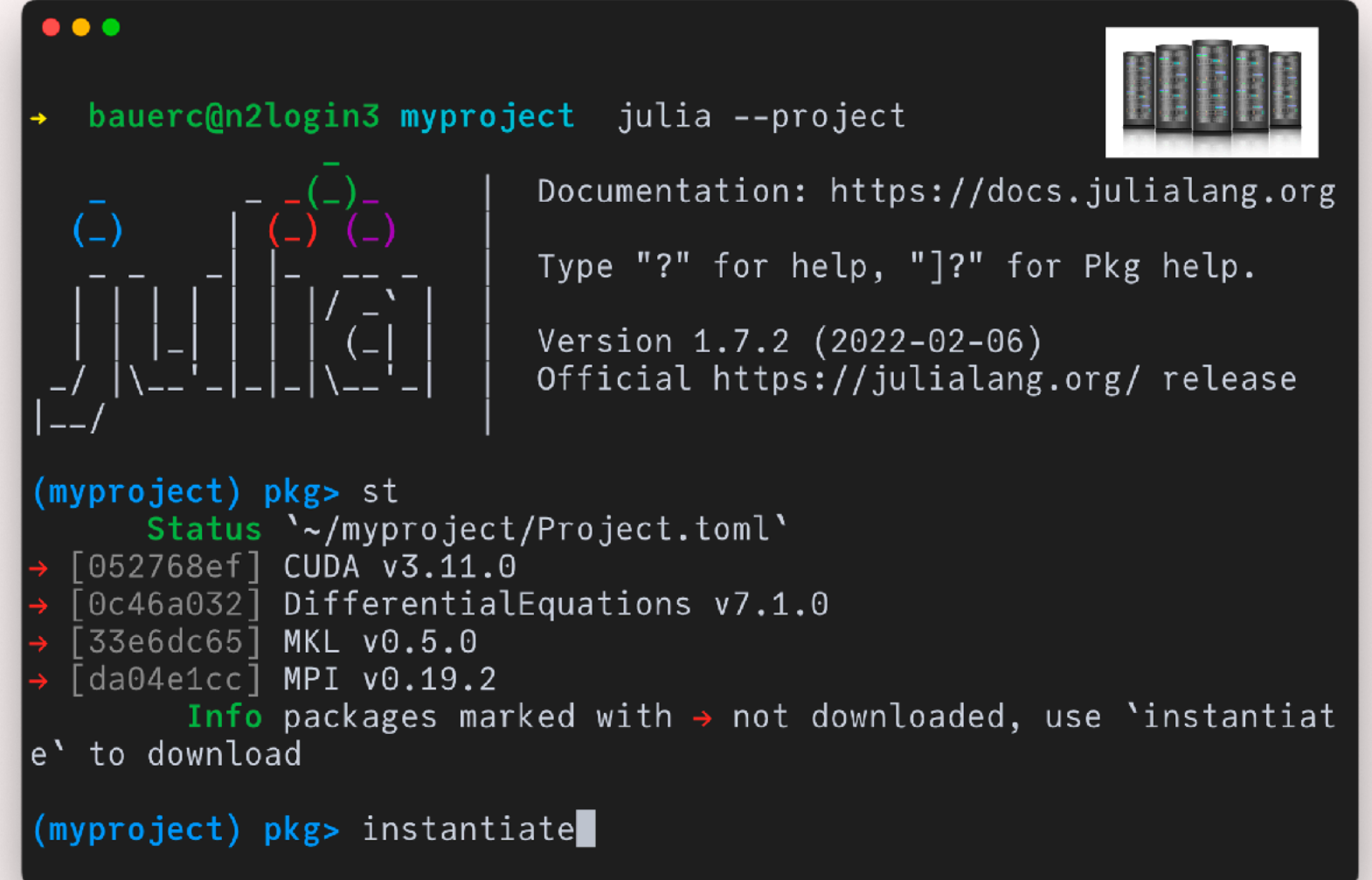
```
→ ~/myproject tree
.
├── Manifest.toml
├── Project.toml
└── code.jl

0 directories, 3 files

→ ~/myproject cat Project.toml
[deps]
CUDA = "052768ef-5323-5732-b1bb-66c8b64840ba"
DifferentialEquations = "0c46a032-eb83-5123-abaf-570d42b7fbba"
MKL = "33e6dc65-8f57-5167-99aa-e5a354878fb2"
MPI = "da04e1cc-30fd-572f-bb4f-1f8673147195"

→ ~/myproject █
```

HPC Cluster



```
→ bauerc@n2login3 myproject julia --project

Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.
Version 1.7.2 (2022-02-06)
Official https://julialang.org/ release

(myproject) pkg> st
Status `~/myproject/Project.toml`
→ [052768ef] CUDA v3.11.0
→ [0c46a032] DifferentialEquations v7.1.0
→ [33e6dc65] MKL v0.5.0
→ [da04e1cc] MPI v0.19.2
Info packages marked with → not downloaded, use `instantiate` to download

(myproject) pkg> instantiate█
```

(Using **system software** is supported.)

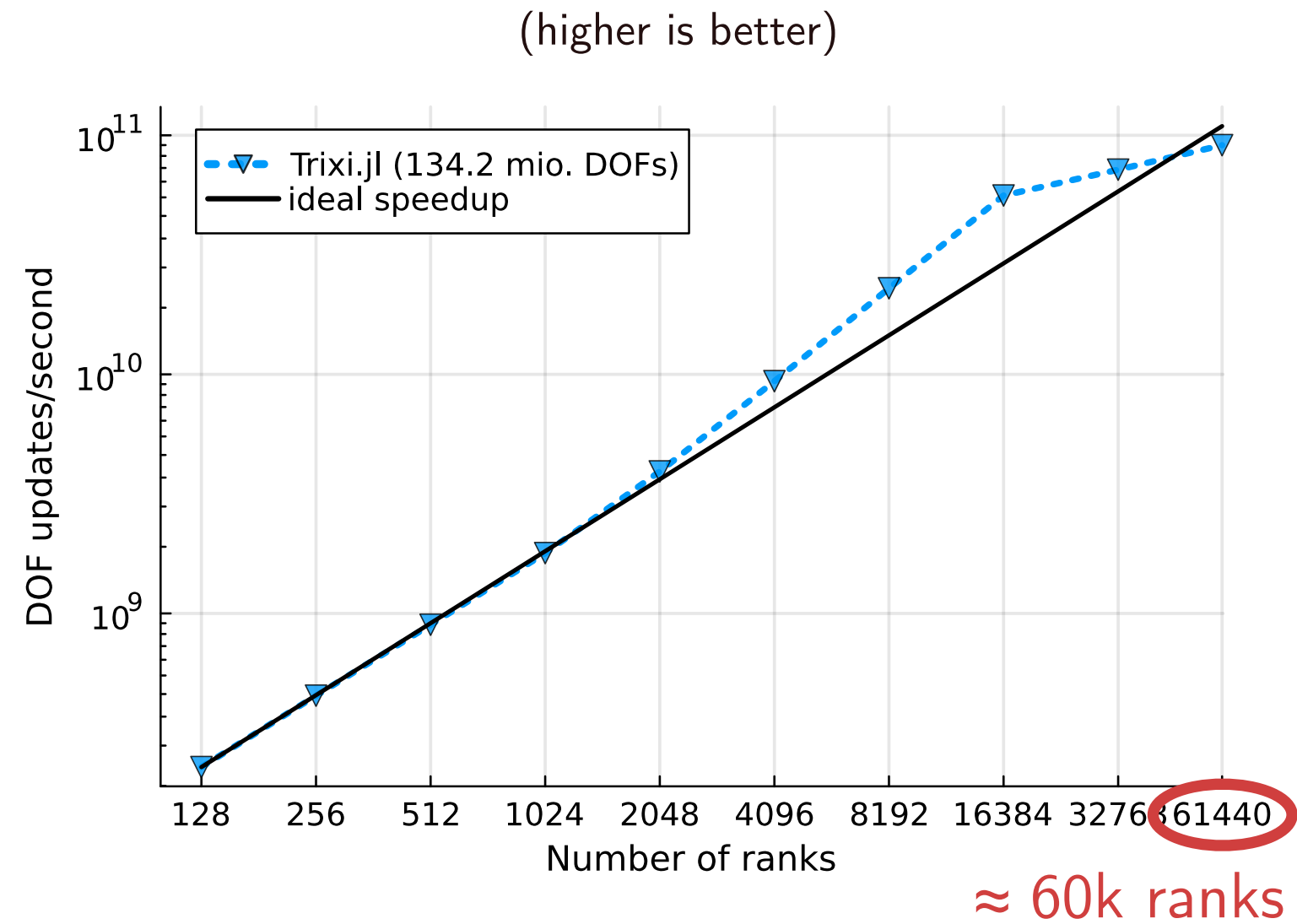
Julia code can be  
**fast** and **scalable**.

Type inference

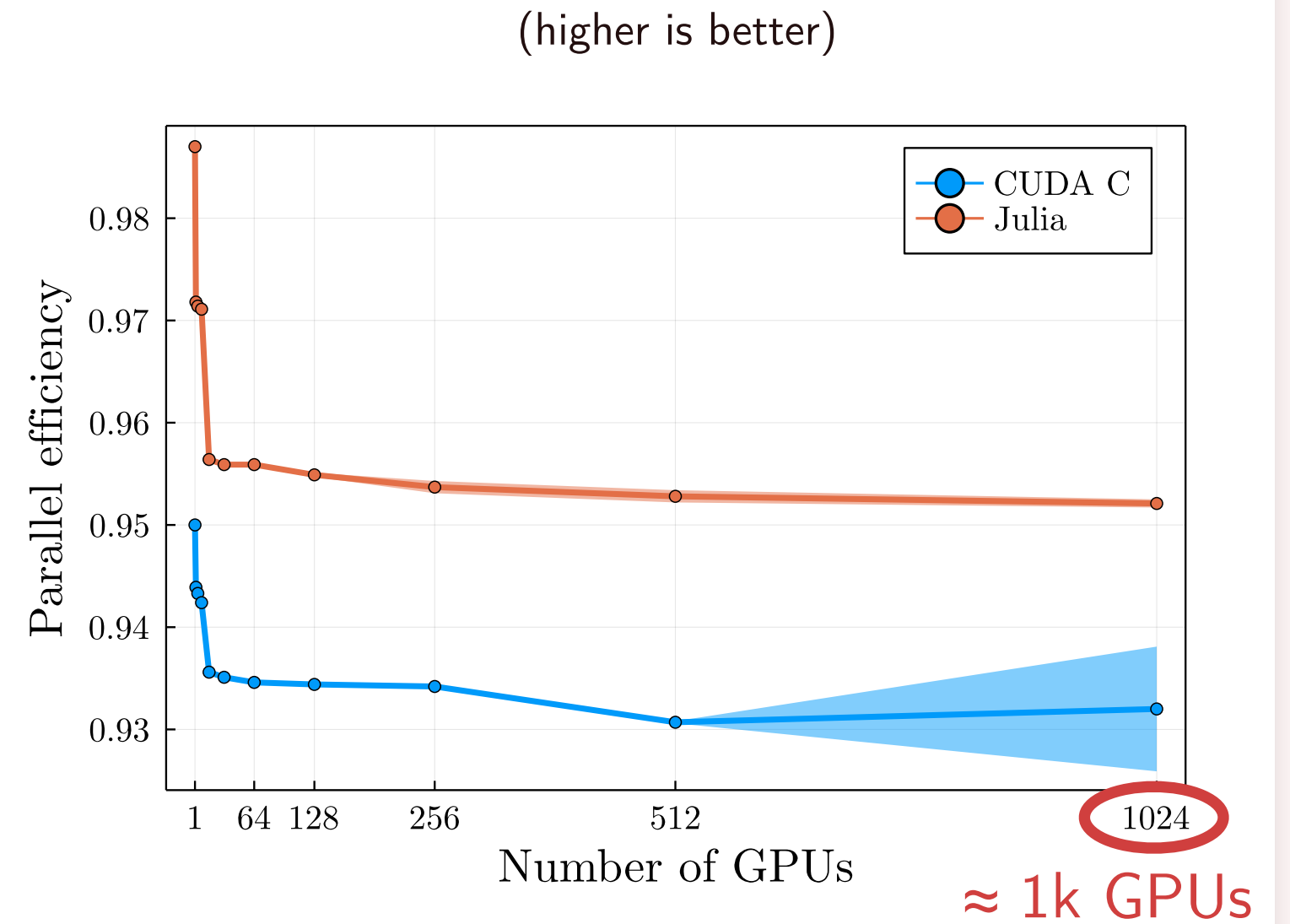
Compilation via **LLVM**

**MPI** support

# Good scaling of PDE codes



Trixi.jl (Multi-CPU)



ParallelStencil.jl (Multi-GPU)

Julia invites you to  
gradually **delve**  
**deeper.**

Entirely **open source**

Julia is (mostly) **written in Julia**

Great **introspection tools**

# The Julia HPC Community



A small but vibrant  
and welcoming  
community.

People with passion and drive

International  
(NERSC, ORNL, CSCS, PC2, ...)

Opportunity to join and grow

# Many people are using Julia for HPC.

## **CLiMA** @ Caltech

- Climate modelling

## **Trixi** @ RWTH Aachen / HLRS

- Adaptive simulations of conservation laws

## **GPU4GEO** @ ETH / CSCS

- Computational earth science

## **WaterLily** @ BSC / TU Delft

- Computational fluid dynamics

...



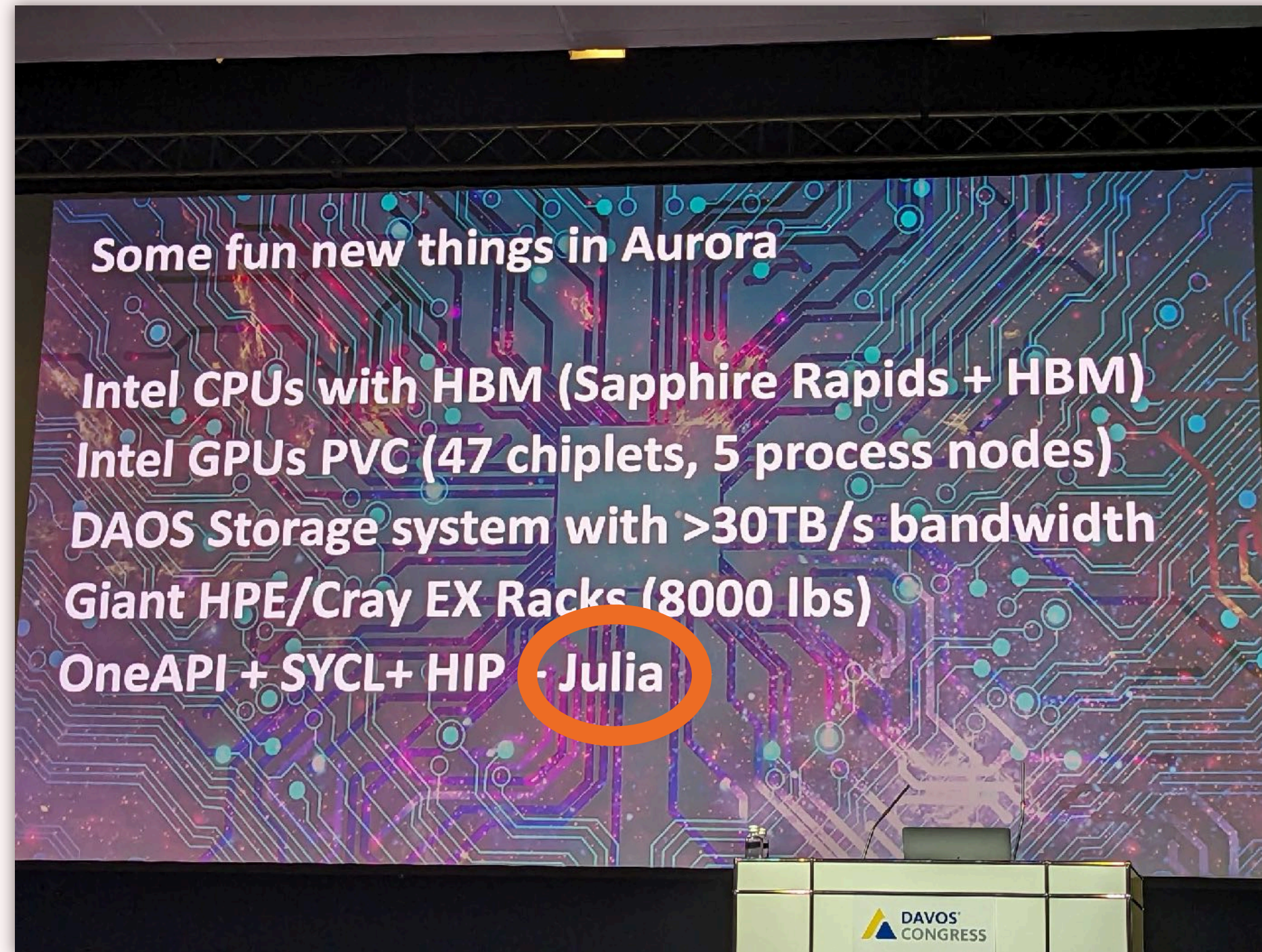
# We welcome you to one of our sessions ...



... or our monthly Zoom call  
(open to everyone!)



# Julia is a “fun new thing” on Aurora (ANL)



Julia has promising potential for HPC,  
and I invite you to join us in exploring  
and developing it.

