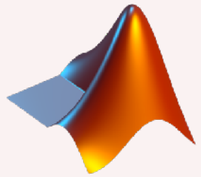
Introduction to Julia for High-Performance Computing

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”

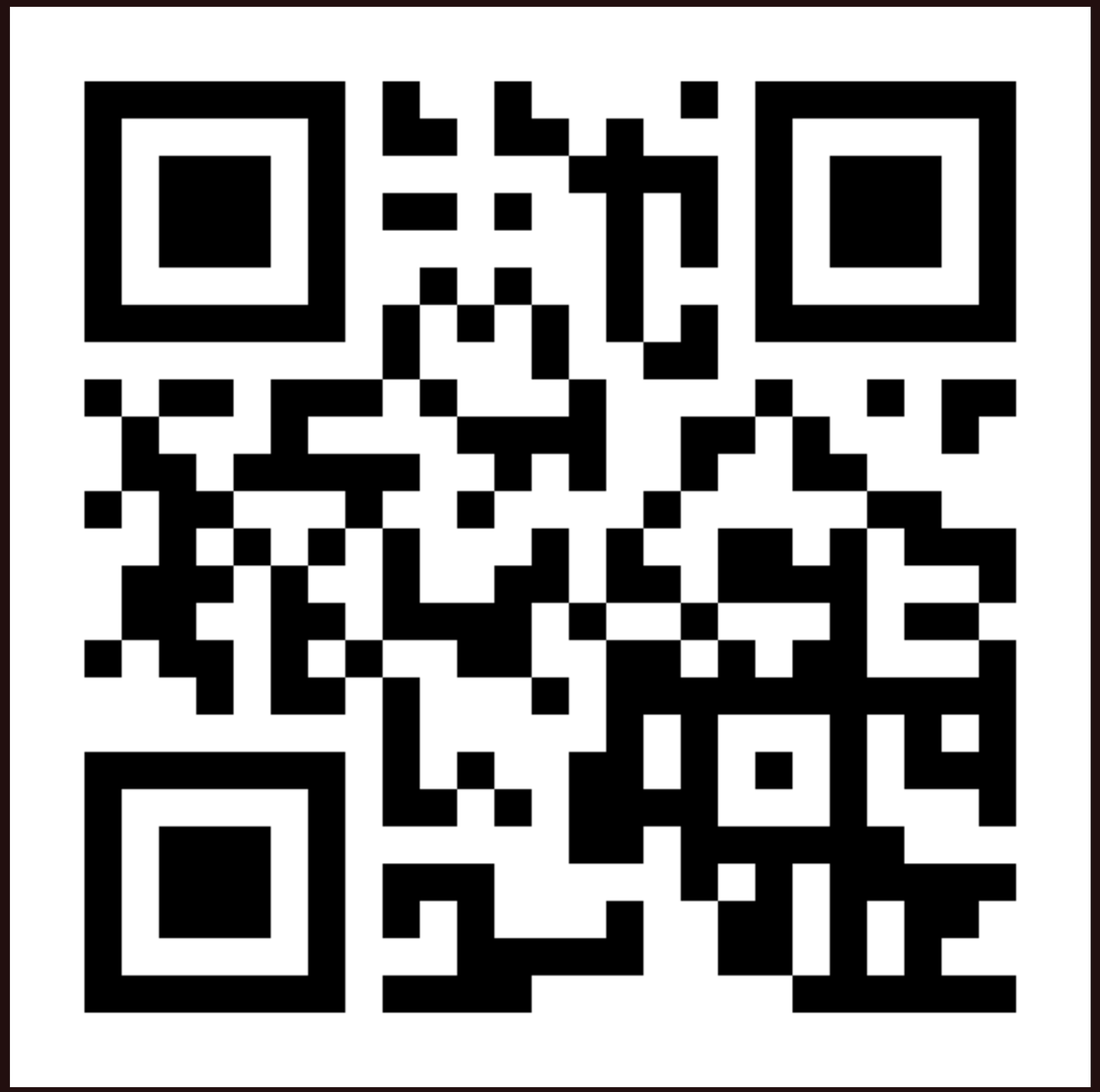
nature

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

	Tuesday	Wednesday	Thursday	Friday
	Foundations	**Core**	**Node**	**Cluster**
09:00 - 10:45	Intro Fundamentals	Optimising Performance I	Multithreading	Distributed Computing
10:45 - 11:00	Break	Break	Break	Break
11:00 - 12:30	Compilation	Exercises	Exercises	Exercises
12:30 - 14:00	Lunch	Lunch	Lunch	Lunch
14:00 - 15:30	Fast & Generic Code	Optimising Performance II	GPU Programming	Exercises
15:30 - 15:45	Break	Break	Break	Outro
15:45 - 17:00	Exercises	Exercises	Exercises	

Quick Survey

https://etc.ch/wZaG



Julia's Strengths

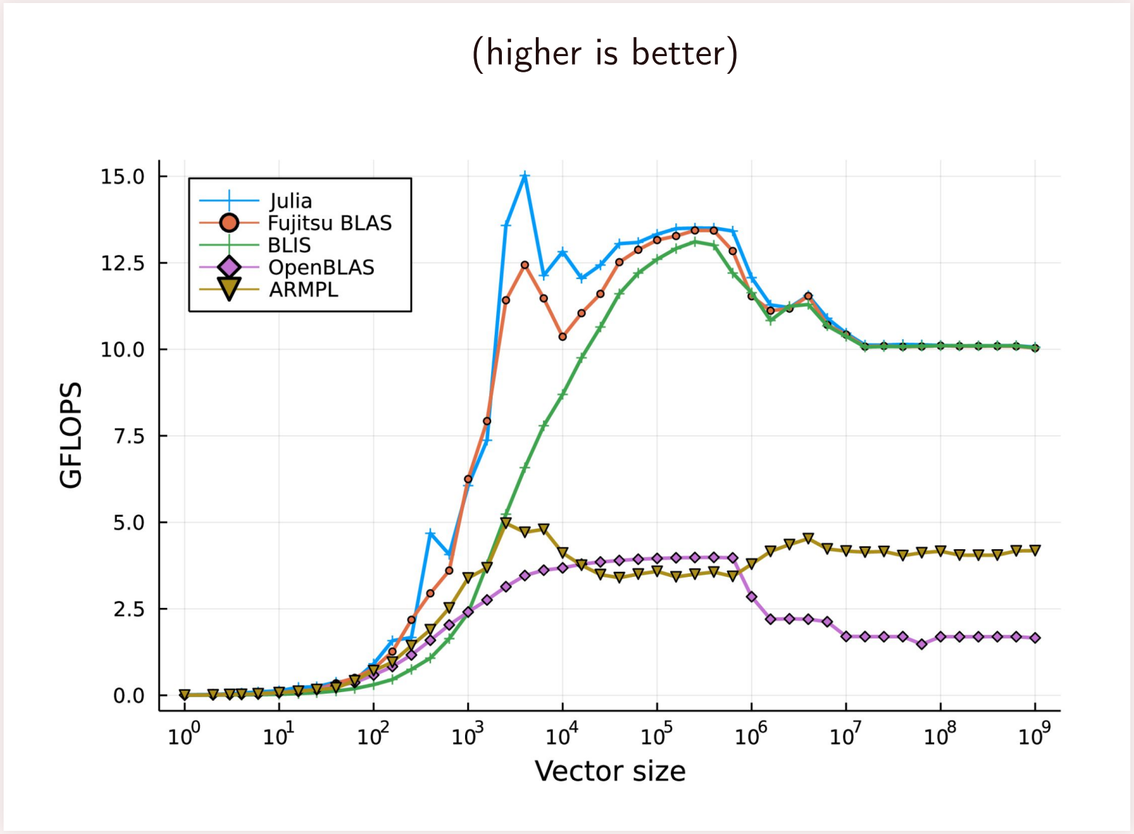
Julia code can be fast and scalable.

Type inference

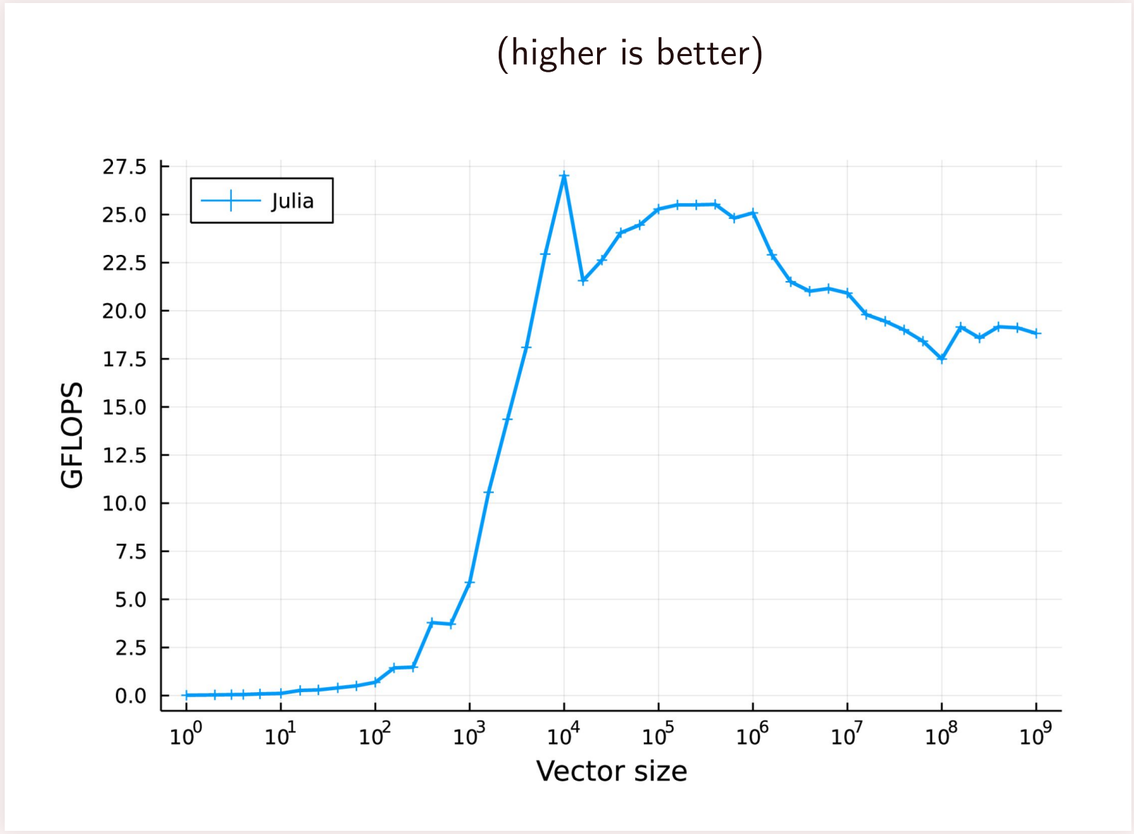
Compilation via LLVM

MPI support

Fast and generic AXPY on Fugaku (A64FX)

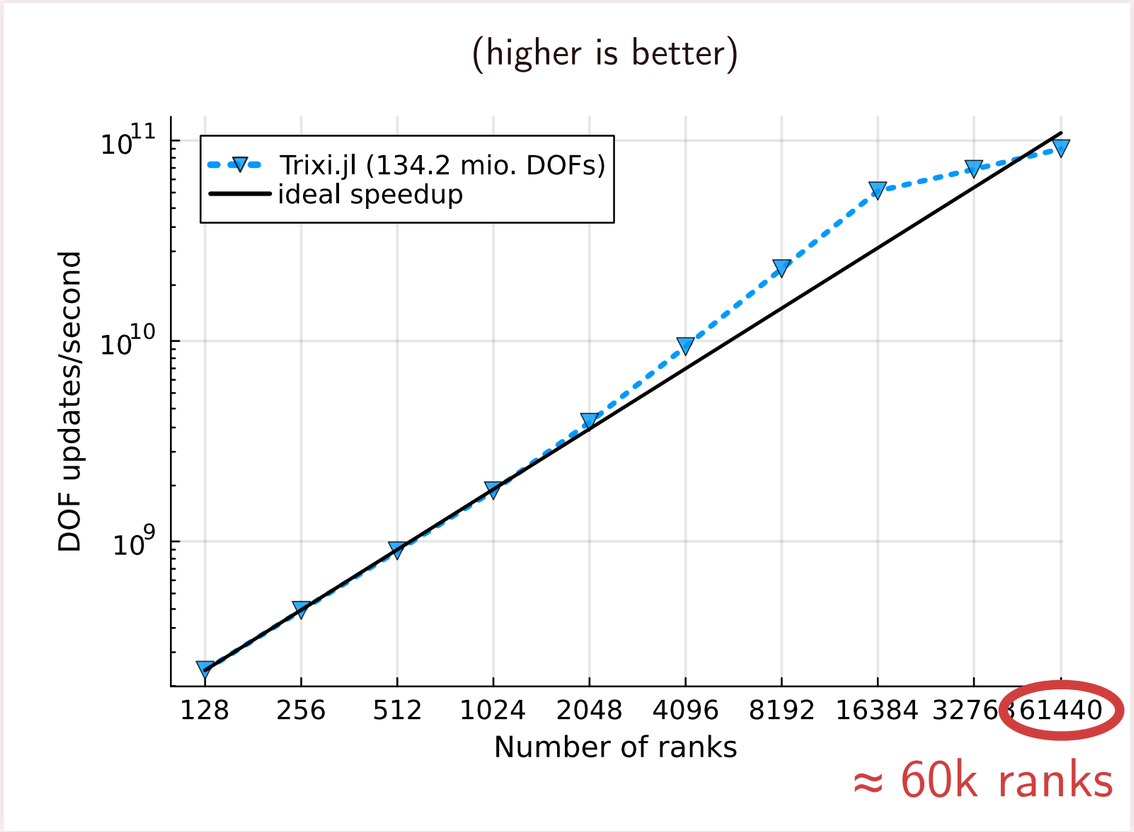


Single-precision (Float32)

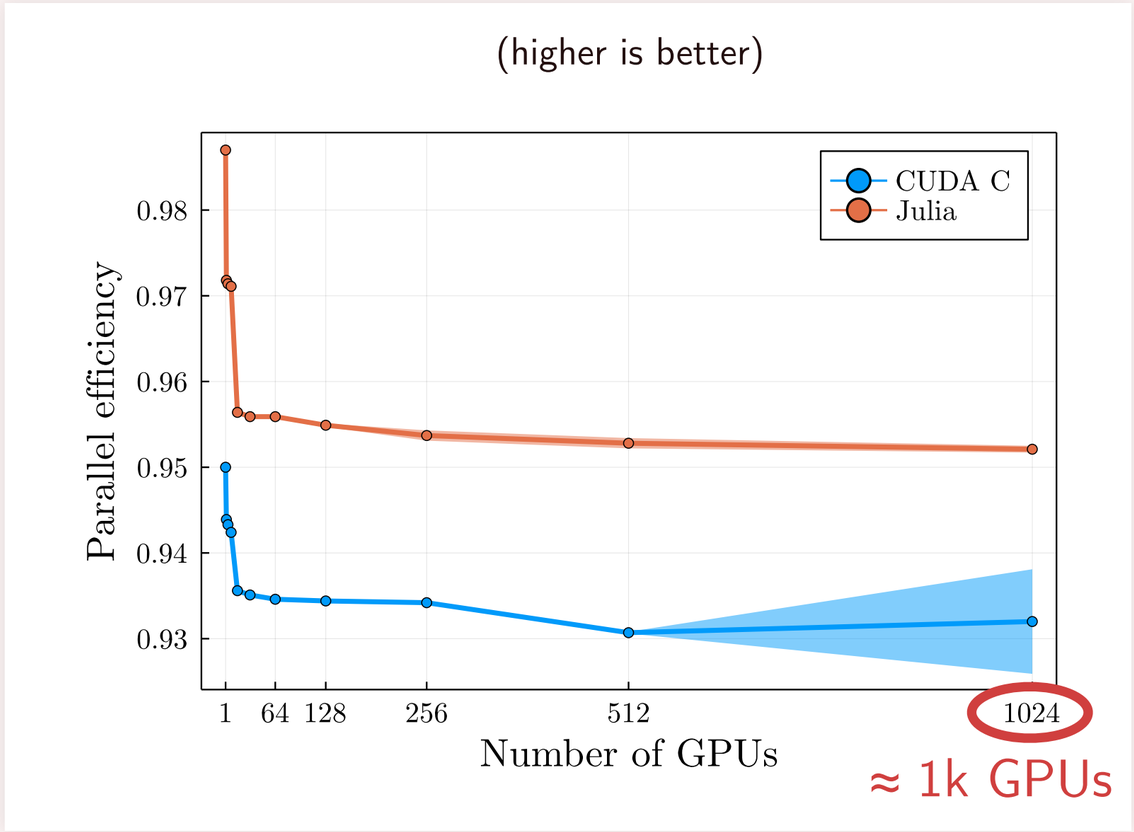


Half-precision (Float16)

Good scaling of PDE codes



Trixi.jl (Multi-CPU)



ParallelStencil.jl (Multi-GPU)

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-570d42b7fbaa"
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 `instantiat
e` to download

(myproject) pkg> instantiate
```

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

Julia invites you to gradually delve deeper.

Entirely open source

Julia is (mostly) written in Julia

Great introspection tools

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

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

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

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