Julia for High-Performance Computing

Panelists:

- William S. Moses – **Auto-Differentiation**
- Pedro Valero-Lara – **Portability**
- Julian Samaroo – **Task-based Parallelism**
- Felipe Tome – **Distributed Computing**
- Steven Hahn – **Julia in Real Application**

Moderator:

- Rabab Alomairy – **Julia for HPC**

Organizers:

- Johannes Blaschke
- William F Godoy
- Mose Giordano

https://github.com/JuliaParallel/julia-hpc-bof-sc24

SC24

Atlanta, GA | hpc creates.

Interactive Session

Please access the SC24 digital platform and navigate to the Julia for HPC BoF session to submit your responses.

We value your input!

Agenda

| Time	Speaker	Title
12:15pm – 12:20pm	Rabab Alomairy (MIT)	Opening (Julia for HPC)
12:20pm – 12:25pm	William S. Moses (UIUC)	Automatic Differentiation and Parallelization in Julia
12:25pm – 12:30pm	Pedro Valero-Lara (ORNL)	Leveraging HPC Meta-Programming and Performance Portability with the Just-in-Time and LLVM-based Julia Language
12:30pm – 12:35pm	Julian Samaroo (MIT)	Dagger.jl
12:35pm – 12:40pm	Felipe Tome (USP)	DaggerMPI.jl
12:40pm – 12:45pm	Steven Hahn (ORNL)	My Scientific Journey to Embracing Julia
12:45pm - 1:00pm	Everyone	Discussion

Discussions moderated as well by Johannes Blaschke (LBNL), William F Godoy (ORNL), and Mose Giordano (UCL).

Julia Solves HPC Language Barrier

- HPC accelerates scientific discovery
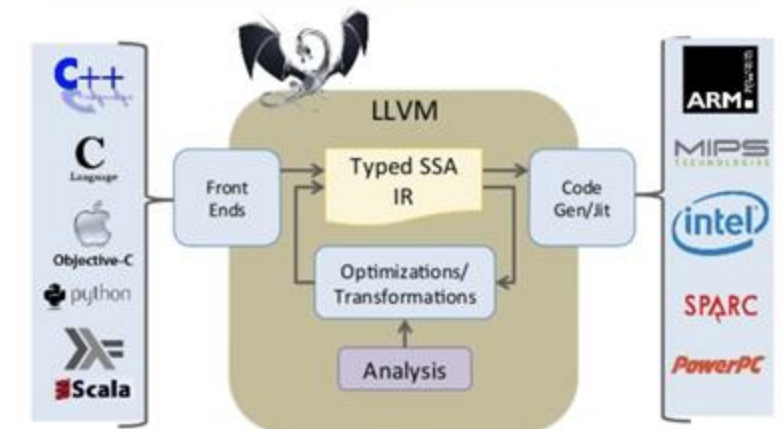
Domain Science ←→ HPC Expert

Language Barrier
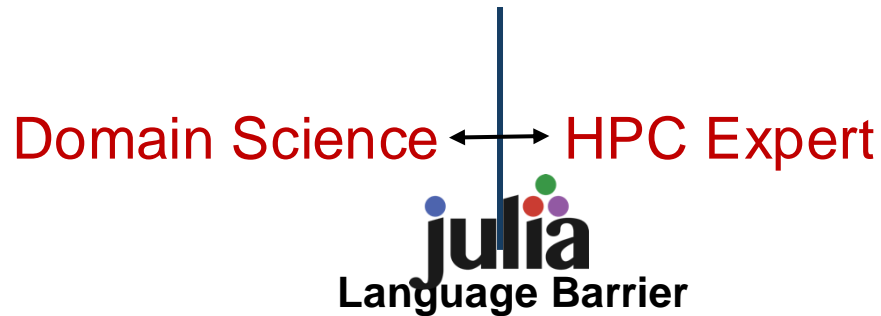
- Julia offers the **high-level productivity** and ease of use that domain scientists love, while delivering the **low-level performance** expected in HPC.

- Julia's secret sauce is its use of **LLVM** JIT compilation.

- Julia generate optimized machine code for diverse architectures.

JuliaHub
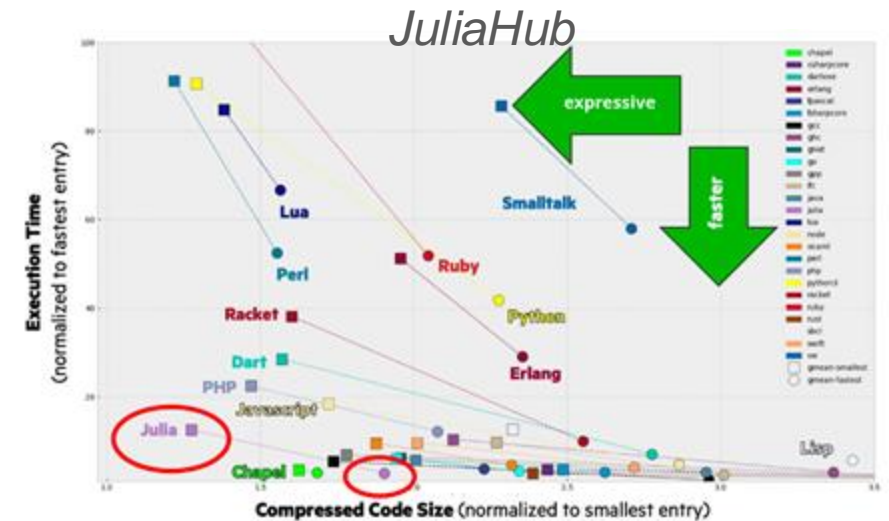
LLVM is an industry-standard compiler infrastructure

Rich Ecosystems for HPC

- Julia has rich ecosystem for HPC: extensive support for parallel computing: Multi-threading, Distributed computing, and GPU acceleration.

Julia Seamless Integration

- Julia provides seamless integration with existing HPC infrastructure:
  - Compatible with job schedulers like SLURM, container orchestration platforms like Kubernetes, and other HPC tools.
  - Easy interoperability with C/C++ and Fortran libraries, allowing incremental adoption.

Julia for High-Performance Computing

- What are the TECHNICAL aspects or features you like MOST about Julia?
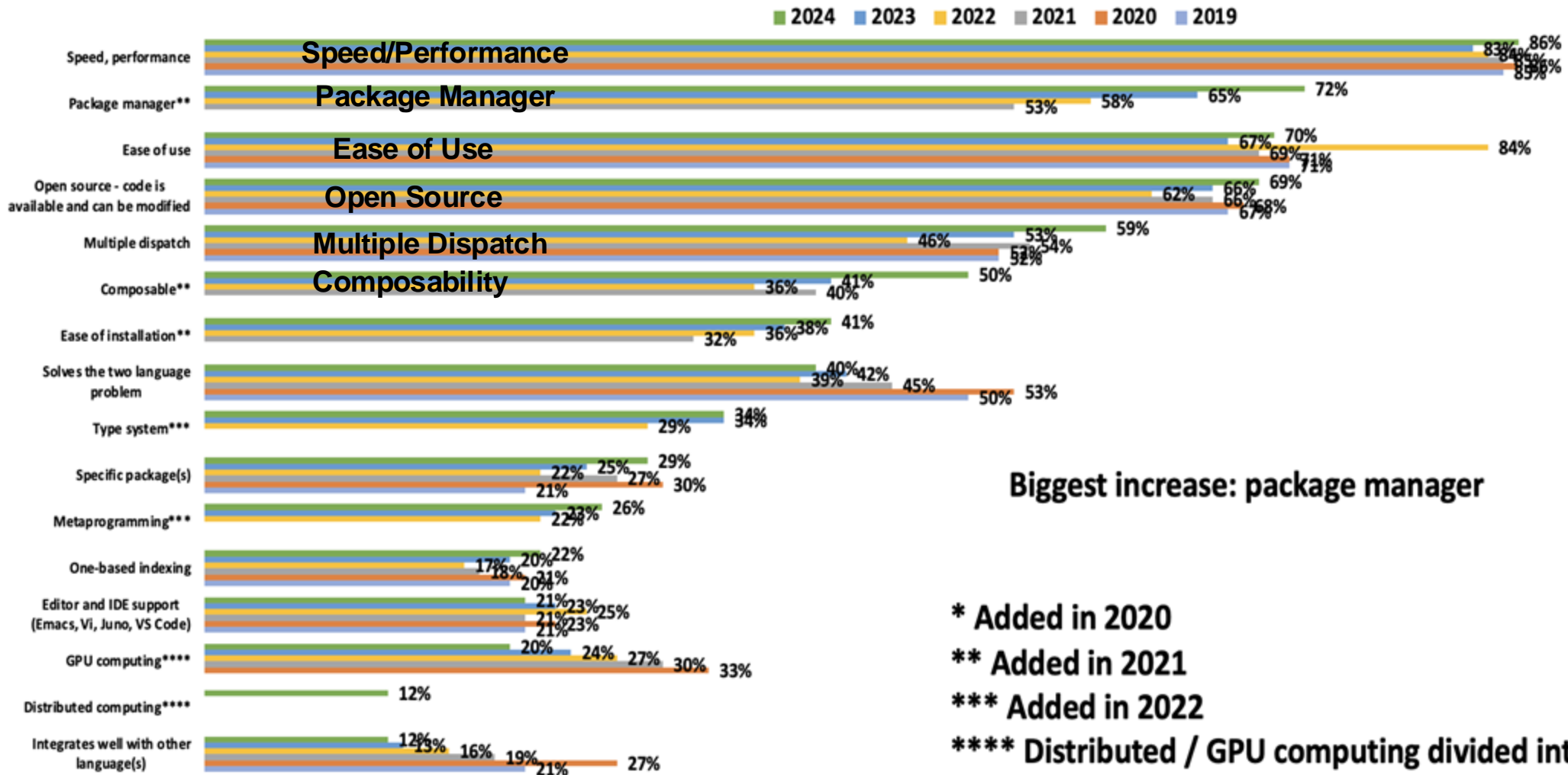
Julia Community and Adoption

- Julia has growing community: user base has been expanding in scientific computing, machine learning, and AI
- Julia's adoption continues to grow rapidly across academia, research labs, and industry
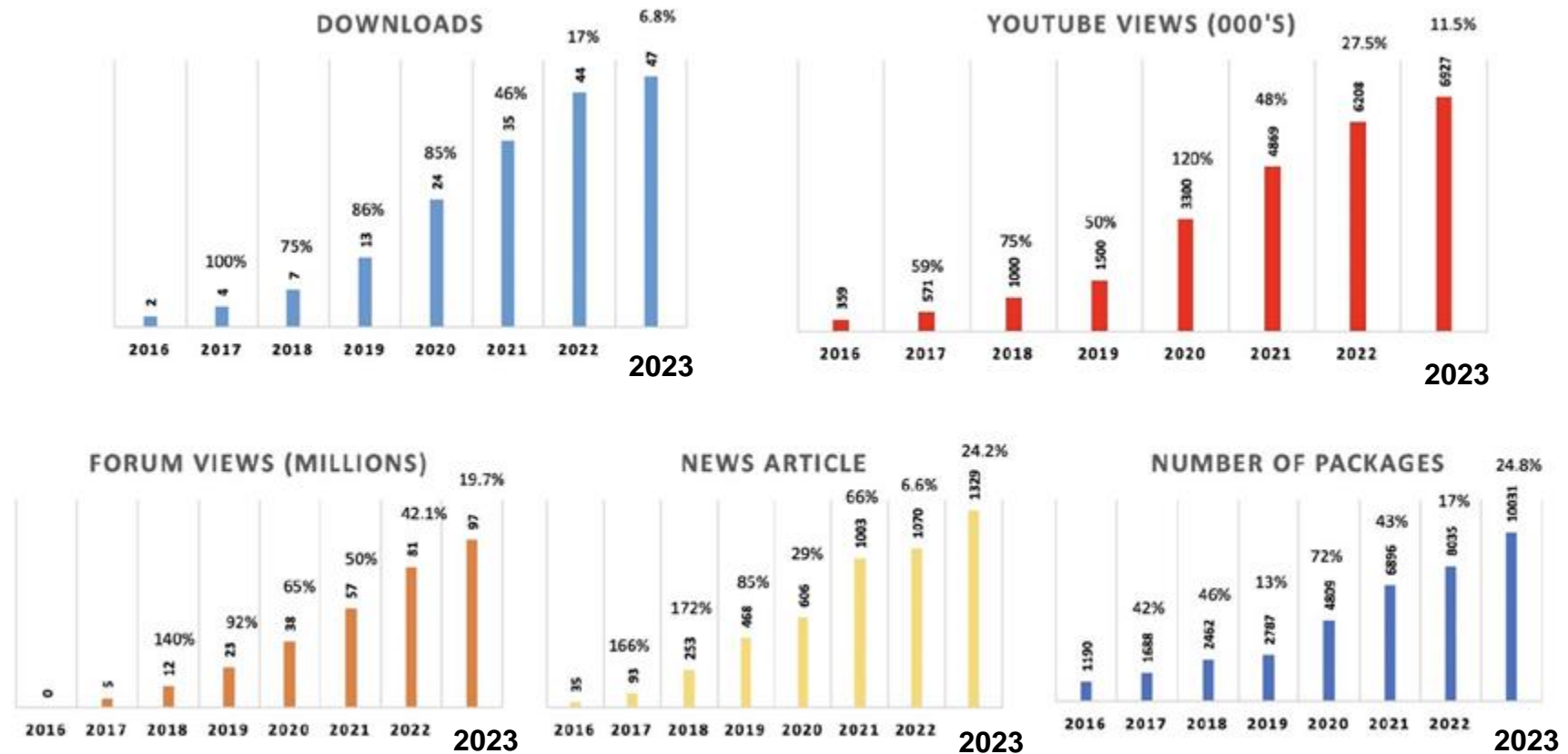
Rabab Alomairy

Julia Packages
Dagger.jl
DLA.jl

Introduction
Postdoctoral at MIT JuliaLab

Research Interest
- High Performance Computing
- Approximate Computing
- Task-based Runtime
- Designing Scalable Linear Algebra Algorithms

Why Julia?
- Julia's expressiveness and its ability to effectively utilize modern hardware make it an excellent choice for developing high-performance algorithms and simulations

William S. Moses

Introduction

Assistant Professor at the University of Illinois in the Computer Science and Electrical and Computer Engineering departments.

Research Interest

- High Performance Computing
- Compiler Optimization and Automation
- Automated Differentiation and Program Fusion
- LLVM IR Extensions for Parallel Optimization
- Automated Linear Algebra and Parallelization

Why Julia?

- Honestly it's a combination of both the community and the ease with which one can hack fun program transformations together .

Julia Packages

Enzyme.jl
Reactant.jl

Pedro Valero-Lara

Introduction
Senior Computer Scientist at Oak Ridge National Laboratory

Research Interest
- High Performance and Parallel Computing
- Scientific Computing/Programming
- Software Sustainability, Performance Portability and Programming Productivity
- Generative AI Models for HPC Programming

Why Julia?
Programming productivity for scientific and high-performance computing.

Julia Packages
JACC.jl
GrayScott.jl

Julian Samaroo

Introduction
Research Engineer at MIT's JuliaLab

Research Interest
- GPU and Distributed HPC
- Task runtimes
- ML-driven scheduling
- High-level, User-friendly Parallel Programming Interfaces

Why Julia?
The combination of multiple dispatch, generic interfaces, metaprogramming, a built-in task runtime, and a flexible compiler unlocks infinite possibilities for productive HPC.

Julia Packages
Dagger.jl
AMDGPU.jl

Felipe Tome

Universidade de São Paulo

Julia Packages
Dagger.jl
LinearAlgebra.jl

Introduction

Brazilian Computer Engineering Undergraduate, soon to be a PhD candidate next semester at University of São Paulo

Research Interest

- High Performance Computing
- Large Scale Physical Simulations
- Wave Structuring Simulations
- Task-based scheduler

Why Julia?

The use of Julia made my physical project feasible, in the sense that being the only one currently dealing with this area in acoustics made me search for performant and productive high-level languages.

Steven Hahn

Julia Packages
MiniVATES.jl
JACC.jl

Introduction

Software Scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL).

Research Interest
- High Performance Computing
- Power Grid Analysis
- Neutron Scattering
- GPU Acceleration

Why Julia?

I am intrigued by Julia's value proposition to provide a productive and performant scientific programming language on top of LLVM.

AUDIENCE POLLING

Now is the time for audience through our interactive poll.

Please access the SC24 digital platform and navigate to the Julia for HPC BoF session to submit your responses.

We value your input!

Audience Question: Julia Familiarity

How familiar are you with the Julia programming language for high-performance computing applications?

A) I have extensive experience with Julia.
B) I have used Julia a few times.
C) I am aware of Julia but have not used it.
D) I am not familiar with Julia at all.

Expert Question

Can you highlight what set Julia apart and what features or capabilities you have found most transformative compared to other languages?

Audience Question: Automatic differentiation

To what extent do you agree with the following: "Automatic differentiation tools like Enzyme play a crucial role in hiding the complexity of differentiating GPU kernels, allowing developers to focus on higher-level problem-solving."

A) Strongly agree
B) Agree
C) Neutral
D) Disagree
E) Strongly disagree

The majority agrees:

Great to see strong agreement! Could you elaborate on how Julia helps with this abstraction? What specific features or tools in Julia have been most effective in achieving this, particularly in the context of GPU computing?

Expert Question

The majority is neutral or disagrees:

Why do you think there was less agreement on the role of tools like Enzyme in simplifying differentiation for GPU kernels? What improvements could be made from both a research and software engineering perspective?

Audience Question: Portability

If you were selecting a programming language for a project, how important would portability be in your decision?

A) Very important
B) Somewhat important
C) Neutral
D) Not a consideration

Expert Question

The majority selects "Very important" or "Somewhat important":

Portability is clearly a priority for many of you. How do you see Julia's current design and its integration with LLVM playing a role in supporting emerging specialized AI chips like SambaNova, Cerebras (CS-3 system), and Neural Processing Units (NPUs)?

Expert Question

The majority is neutral or selects "Not a consideration":

In your view, could extending Julia's support to specialized AI chips like SambaNova, Cerebras, and NPUs help shift this perception? How can Julia's design be improved to ensure better adaptability for evolving hardware, and what features could enhance its portability for future architectures?

Audience Question: Portability

What do you see as the biggest challenge for a language aiming to achieve portability across various hardware (e.g., CPUs, GPUs, AI chips)?

A) Maintaining consistent performance across all platforms
B) Supporting specialized hardware features without complicating the code
C) Managing memory and resources efficiently across diverse systems
D) Ensuring compatibility with existing software across different environments

Expert Question

The majority selects "Maintaining consistent performance across all platforms":

Performance consistency seems to be the biggest concern when aiming for portability across different hardware types. Given the diverse memory models, instruction sets, and parallel execution paradigms, what strategies or abstractions in Julia have you found most effective in maintaining good performance while ensuring portability?

Expert Question

The majority selects "Supporting specialized hardware features without complicating the code":

It's clear that supporting specialized hardware features without introducing complexity is a major challenge. How do you see Julia's current abstractions helping in this regard? What specific features or tools have been effective, and what further improvements could help simplify the integration of specialized hardware features?

Expert Question

The majority selects "Managing memory and resources efficiently across diverse systems":

Efficient memory and resource management across different hardware types appears to be a key challenge. In your experience, what strategies or abstractions in Julia have proven most effective in addressing this? How can Julia's design evolve to handle diverse memory models while maintaining high performance?

Expert Question

The majority selects "Managing memory and resources efficiently across diverse systems":

Efficient memory and resource management across different hardware types appears to be a key challenge. In your experience, what strategies or abstractions in Julia have proven most effective in addressing this? How can Julia's design evolve to handle diverse memory models while maintaining high performance?

Expert Question

The majority selects "Ensuring compatibility with existing software across different environments":

Compatibility with existing software across diverse environments is a critical concern. How do you leverage Julia's design and abstractions to ensure seamless integration and compatibility? What features in Julia could be improved to better address this challenge and enhance portability?

Audience Question: Task-based Parallelism

How valuable do you think an asynchronous model like Dagger, Dask, or PARSEC is for helping non-expert users avoid dealing with the complexities of scheduling in a heterogeneous system?

A) Extremely valuable; it allows users to focus on their code without worrying about low-level scheduling details.
B) Somewhat valuable; it reduces complexity but still requires a basic understanding of task distribution.
C) Slightly valuable; non-expert users still need to understand the underlying hardware for optimal performance.
D) Not valuable; users will need to manage scheduling manually for the best results in complex systems.

Expert Question

How does Dagger.jl's asynchronous execution model effectively abstract away the complexities of task scheduling and resource management, particularly in heterogeneous hardware environments? What key features of Dagger.jl make it possible for users to focus on high-level code development without getting bogged down by the low-level details of the underlying hardware?

Audience Question: Productivity

In your opinion, how will using Julia for scientific computing impact domain scientists' productivity?

A) It will significantly increase productivity due to rapid prototyping capabilities.

B) It will have a moderate impact, mainly in areas requiring performance tuning or certain Julia-specific features.

C) It may have minimal impact unless there is more training and community support.

D) It could slow down development due to the learning curve and having to support another language.

Expert Question

The majority selects: significantly increase productivity or will have a moderate impact

One of Julia's key strengths is its ability to provide both high-level productivity and low-level performance. Could you share an insight of how Julia helped you achieve this balance, particularly in real application? What features or tools made the biggest difference in streamlining your development process?

Expert Question

The majority selects: minimal impact or slow down development

It seems there are concerns about Julia's adoption and its potential impact on productivity, especially due to the learning curve and the need for more support. Do you believe that Julia's capabilities could eventually drive broader adoption among domain scientists, and if so, what improvements or additional training resources would be needed to make this transition smoother, particularly for complex application?

Please fill out our Survey. It will only take a minute