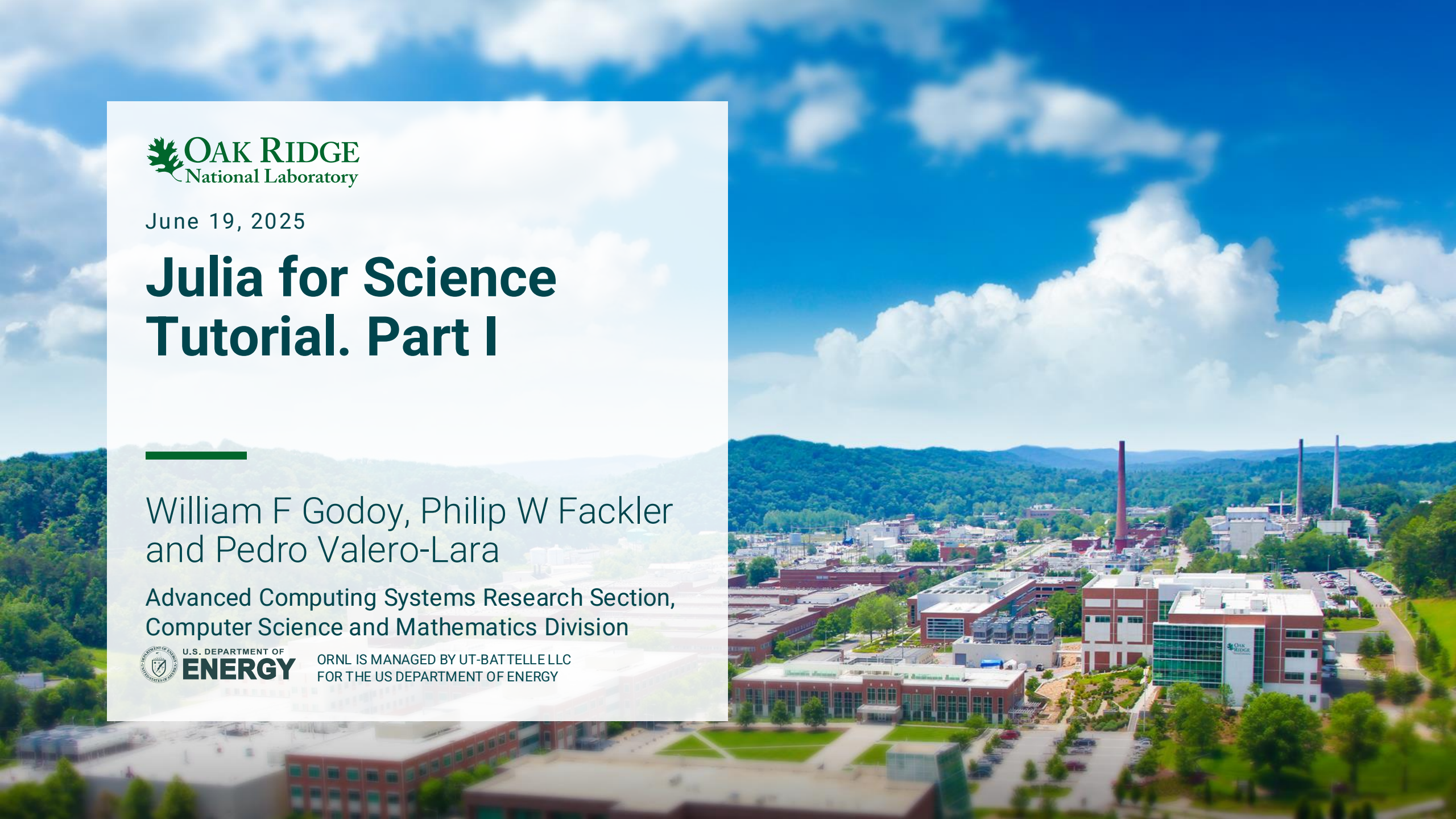


OAK RIDGE National Laboratory

June 19, 2025

Julia for Science Tutorial. Part I

William F Godoy, Philip W Fackler and Pedro Valero-Lara

Advanced Computing Systems Research Section, Computer Science and Mathematics Division

U.S. DEPARTMENT OF ENERGY

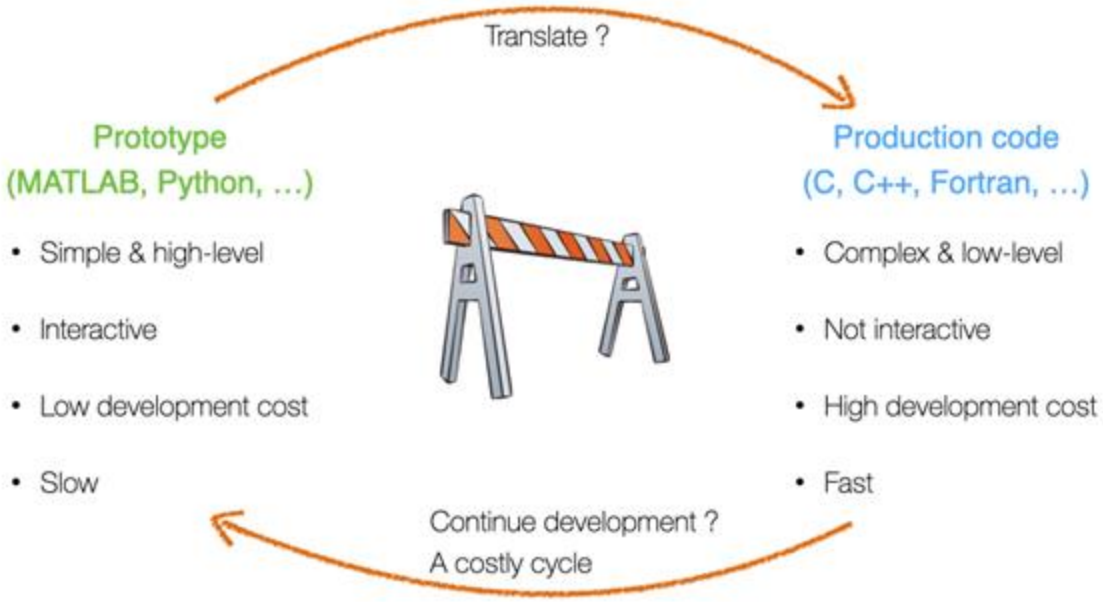
ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

What to expect

- **Day 1:**
  - **Intro to Julia**
  - **Intro to Parallel Programming in Julia**
  - **Intro to JACC (performance portability)**
- **Day 2:**
  - **Running on OLCF systems (Odo and JupyterHub)**
  - **Julia for HPC: JACC, MPI.jl, ADIOS2.jl**

Future events:

- Tutorials at JuliaCon, ICPP, eScience
- SC25 Tutorial and BoF submissions, after SC24 events

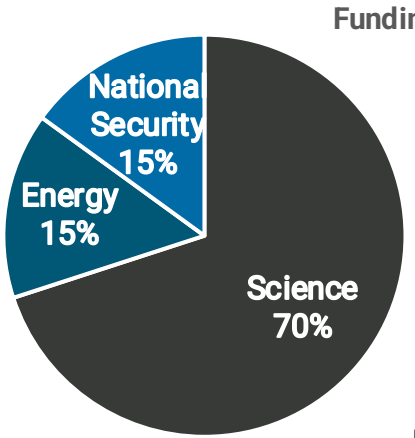


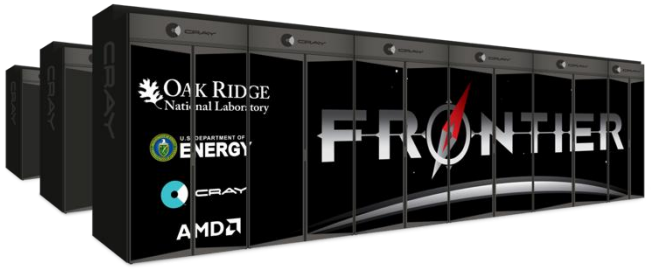
https://pde-on-gpu.vaw.ethz.ch/lecture7

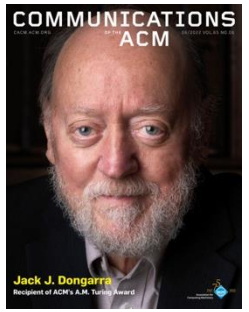
https://scientificcoder.com/my-target-audience

A bit about:
Oak Ridge National Laboratory

- US Department of Energy Office of Science Lab
- Budget $2.4 billion
- 7K staff
- More than 100 disciplines
- 3,200 facility users and visiting scientists
- User facilities
  - Oak Ridge Leadership Computing (OLCF)
  - Building Technologies Research and Integration Center (BTRIC)
  - Carbon Fiber Technology Facility (CFTF)
  - Center for Nanophase Materials Sciences (CNMS)
  - Center for Structural Molecular Biology (CSMB)
  - High Flux Isotope Reactor (HFIR)
  - Manufacturing Demonstration Facility (MDF)
  - National Transportation Research Center (NTRC)
  - Spallation Neutron Source (SNS)







Great place for a scientific career!

Immediate “scientific” (Fortran-like) access to LLVM

```
julia> function add(x,y)
          return x+y
       end

julia> @code_llvm add(2,3)
;  @ REPL[1]:1 within `add`
define i64 @julia_add_132(i64 signext %0, i64 signext %1) #0 {
top:
;  @ REPL[1]:2 within `add`
; ┌ @ int.jl:87 within `+`
   %2 = add i64 %1, %0
; └
  ret i64 %2
}
```

https://godbolt.org/

17 July, 2023

Newsletter July 2023 - JuliaHub Receives $13 Million Strategic Investment from Boeing-Backed AEI HorizonX

Written by JuliaHub

FUSION

General Atomics releases FUSE—an open-source fusion power design tool

Thu, Oct 17, 2024, 4:01PM | Nuclear News

Earlier this month, General Atomics made its Fusion Synthesis Engine (FUSE) software available to others who want to design and build magnetic confinement fusion power plants.

CUDA Zone

CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

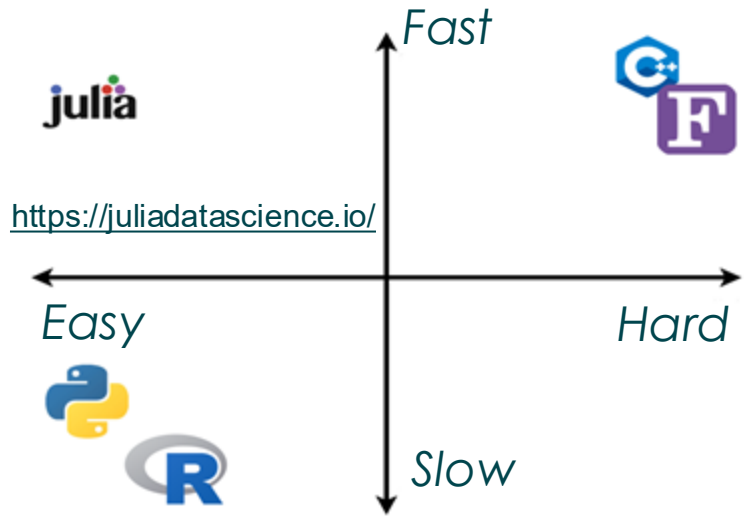
In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python, Julia and MATLAB and express parallelism through extensions in the form of a few basic keywords.

The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.

Download Now

A Heilmeier Catechism for Julia

1. ***What is Julia trying to do?***
   - Close gaps between **productivity** (Python, R) and **performance** (C++, Fortran) and **portability for science**
2. ***How is it done today and what are the limits?***
   - **Python: slow** and **fragmented, C++: complex, Fortran -> LANL report**
   - Complicated ecosystems (packaging, CMake, libraries, etc.)
   - Computing is hard outside x86-64 CPU/NVIDIA GPU
3. ***What is new in your approach and why do you think it will be successful?***
   - Julia is **LLVM** + rich open-source **ecosystem** + **community** for **science**
   - **Investments** in industry, e.g. JuliaHub, Boeing, General Atomics, academia and government
4. ***Who cares? And what difference will it make?***
   - **Scientists** would use one ecosystem = lower costs, accessible hardware (CPU/GPU), **more focus on science**
5. ***What are the risks?***
   - Sustainability and "critical-mass" in 10, 20, 30 years…
   - More popular options: e.g. Mojo, Python+X: JAX, PyTorch?



An evaluation of risks associated with relying on Fortran for mission critical codes for the next 15 years
Shipman, Galen M. Randles, Timothy C.

https://scientificcoder.com/my-target-audience

Landscape of computing: The Tower of Babel

- Scientific software **is divided by**:
  - **Performance languages**: C, C++, Fortran
  - **High-level productivity languages:** Python, Matlab, R

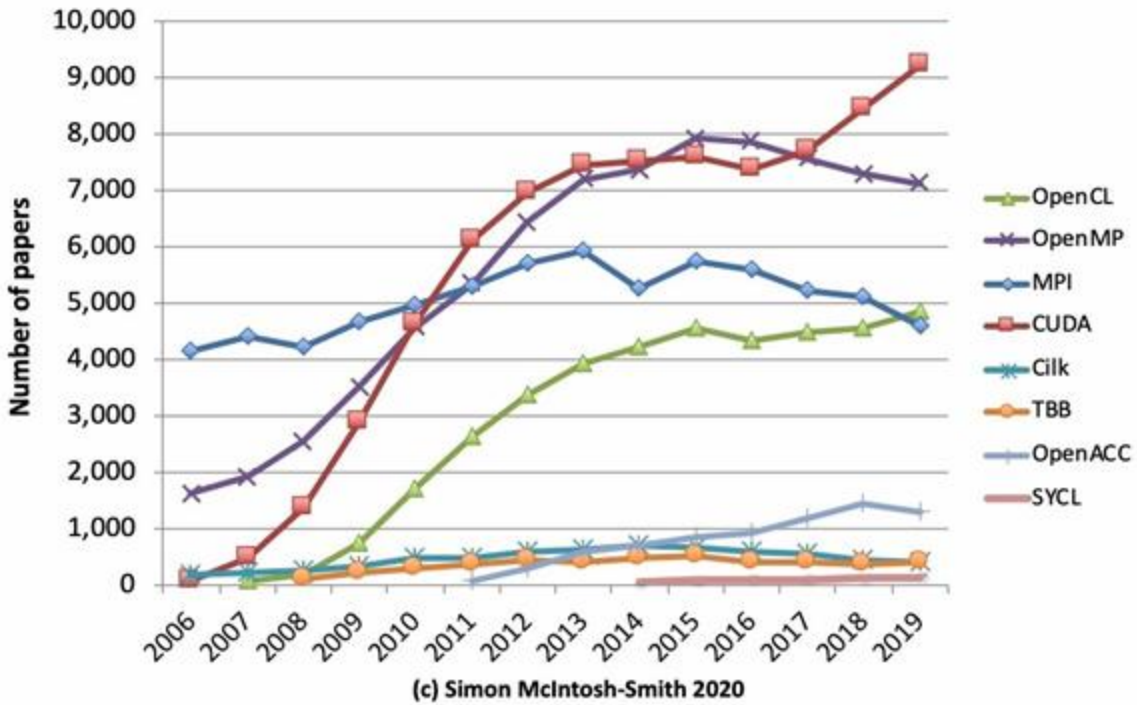
- **Plethora of programming models** for heterogeneous computing . Standard, vendor-specific, third party

- Major shift: vendor compiler convergence around **LLVM**

- **Slowdown in Moore's Law** cadence puts more focus on massively parallel, vectorized computing: Arm CPUs, NVIDIA/AMD/Intel GPU, AI accelerators, TPUs, etc.

- AI (industry) + traditional HPC requires powerful reproducible programming abstractions for computation, communication and data

- Choosing a programming model is not always a technical decision



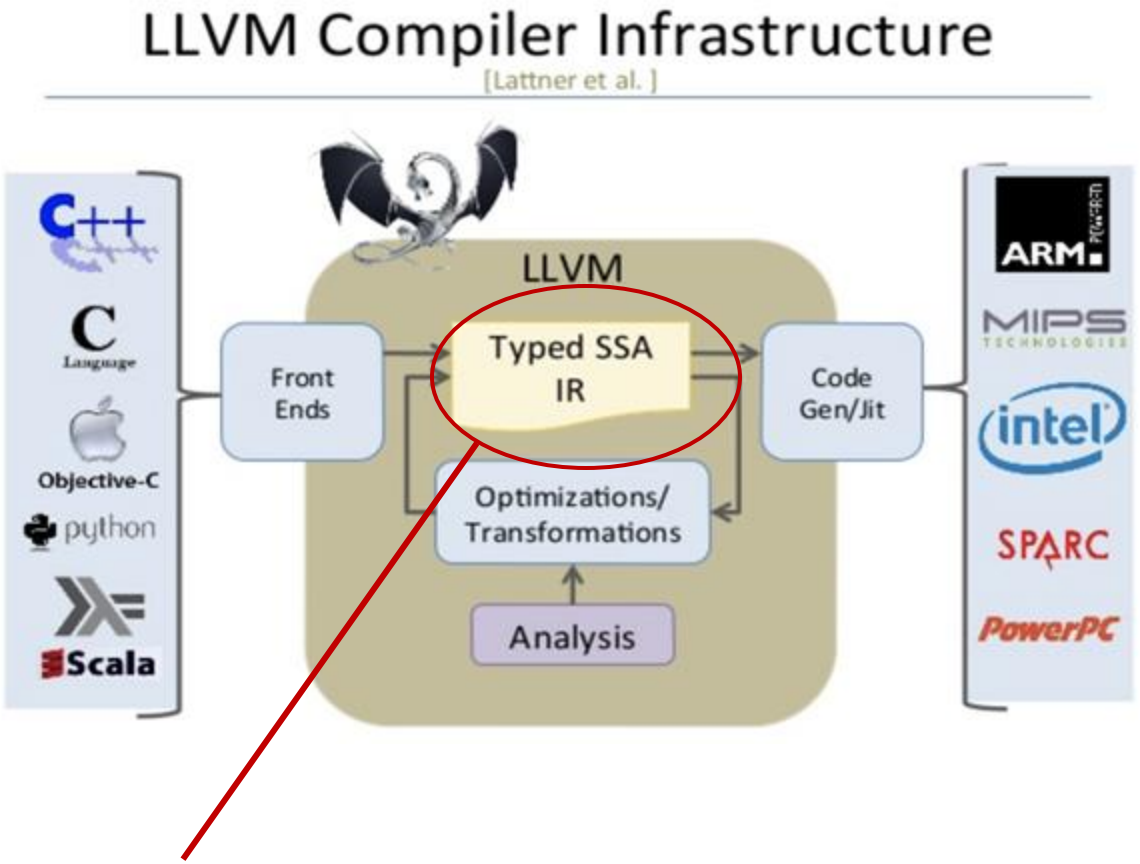
https://www.nextplatform.com/2020/04/28/programming-in-the-parallel-universe/

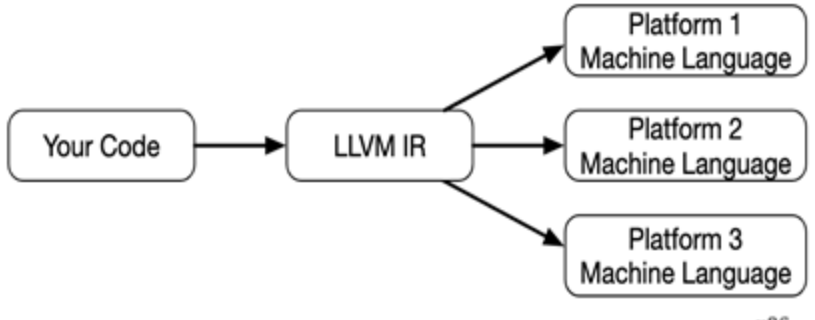
LLVM: a game changer

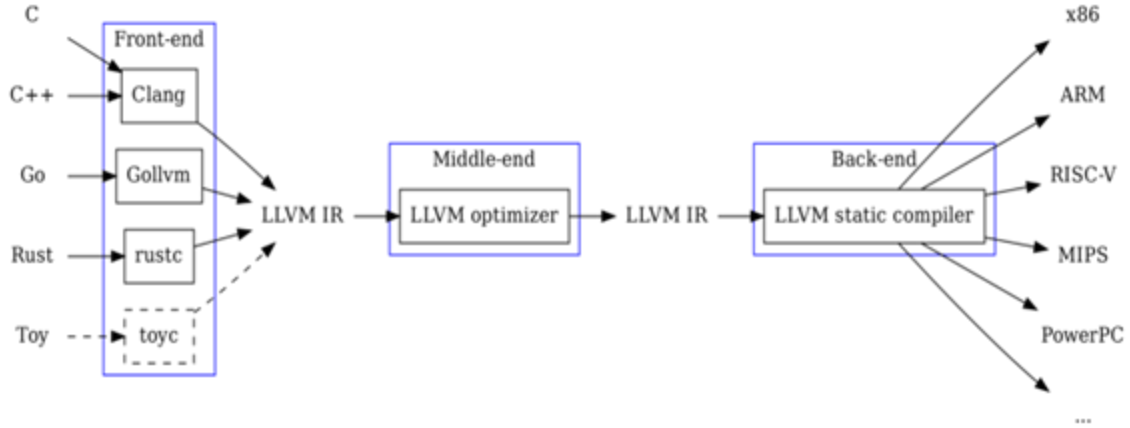
https://llvm.org/

http://www.aosabook.org/en/llvm.html

C. Lattner and V. Adve, "LLVM: a compilation framework for lifelong program analysis & transformation," *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, 2004, pp. 75-86, https://doi.org/10.1109/CGO.2004.1281665 .







LLVM Typed Static Single Assignment (SSA) Intermediate Representation (IR) aka LLVM-IR: https://patshaughnessy.net/2022/2/19/llvm-ir-the-esperanto-of-computer-languages

%57 = call %"Array(Int32)"* @"*Array(Int32)@Array(T)::unsafe_build:Array(Int32)"(i32 610, i32 2), !dbg !89

LLVM: Vendor and programming model adoption

Intel

Developer / Tools / oneAPI / Components / Intel® oneAPI DPC++/C++ Compiler / Intel® Compilers Adopt LLVM

Intel C/C++ compilers complete adoption of LLVM

The next generation Intel C/C++ compilers are even better because they use the LLVM open source infrastructure.

oneAPI

SYCL

https://www.ornl.gov/project/proteas-tune

ARM

ARM Compiler Builds on Open Source LLVM Technology

April 08, 2014

Velocity of open source Clang and LLVM combined with the stability of commercial products improve code quality, performance and power efficiency on ARM processors

arm

OpenACC

Directives for Accelerators

https://csmd.ornl.gov/project/clacc

OpenMP

https://openmp.llvm.org

NVIDIA

NVIDIA CUDA 4.1 Compiler Now Built on LLVM

By Chris Lattner

Dec 19, 2011

CUDA LLVM Compiler

NVIDIA's CUDA Compiler (NVCC) is based on the widely used LLVM open source compiler infrastructure. Developers can create or extend programming languages with support for GPU acceleration using the NVIDIA Compiler SDK.

NVIDIA CUDA

Apple

Fast and powerful

From its earliest conception, Swift was built to be fast. Using the incredibly high-performance LLVM compiler technology, Swift code is transformed into optimized machine code that gets

AMD

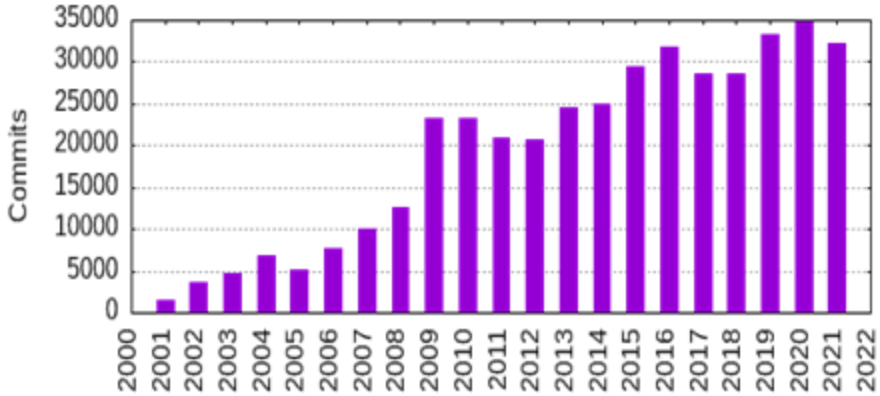
AMD Updates ROCm™ For Heterogenous Software Support

Community support continues to grow for AMD Radeon Open eCosystem (ROCm™), AMD's open source foundation for heterogenous compute. Major development milestones in the latest update include:

- The HIP-Clang compiler is now up-streamed and reviewed by the LLVM™ community, providing a better open source experience for the developer,

AMD ROCm

LLVM Commits

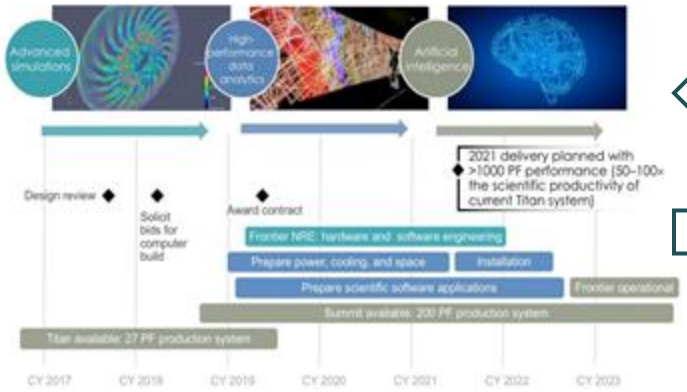


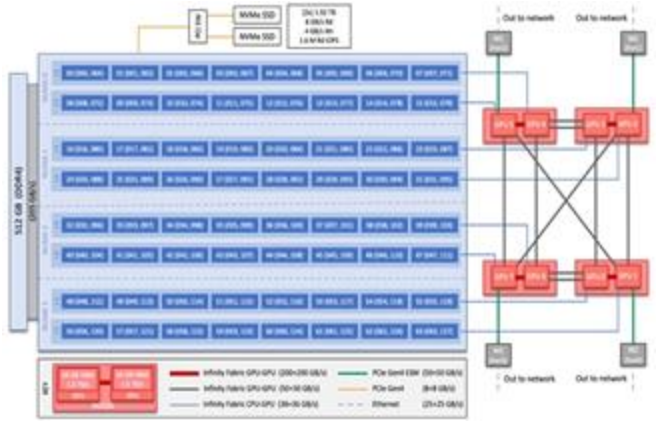
Rethink how we do Computing

- Scientific programming is **HARD** (specially on our Leadership Computing Facilities, LCFs)
- Software is our "specialized science equipment" for science that needs to be **stewarded** and **advanced**

- Programming productivity is always a challenge
- Barrier to entry from idea to portable performance
- Integrate AI (industry)+HPC (government)
- How to leverage legacy codes?

*"Can a machine **translate** a sufficiently rich mathematical language into a sufficiently economical program at a sufficiently **low cost** to make the whole affair feasible?"---------- ---------- Backus on Fortran (1980)*

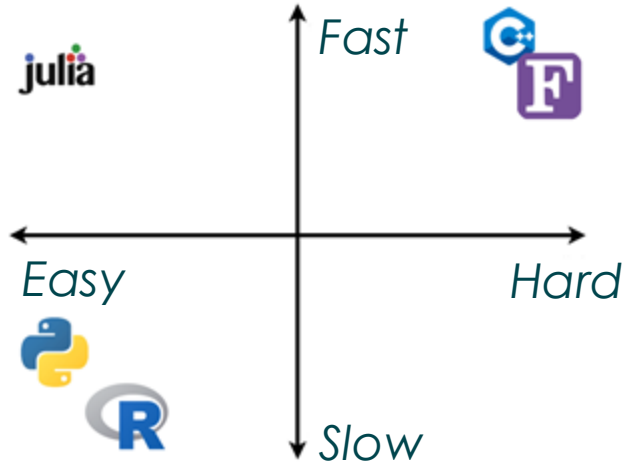




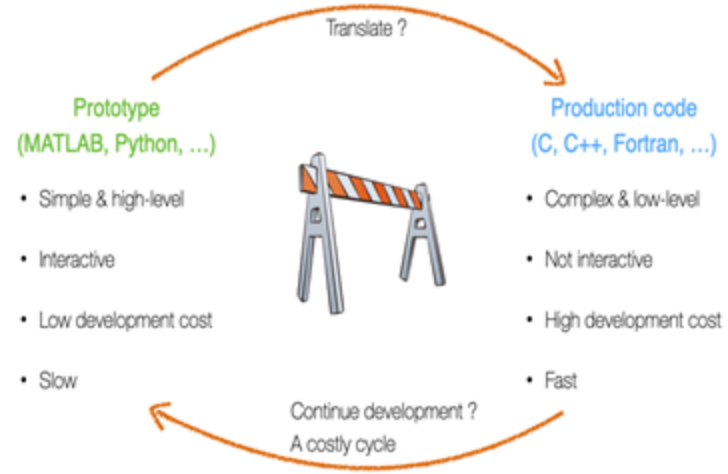
Key question: "What novel approaches to software design and implementation can be developed to provide performance portability for applications across radically diverse computing architectures?" from Reimagining Codesign for Advanced Scientific Computing: Unlocking Transformational Opportunities for Future Computing Systems for Science. DOE Report https://doi.org/10.2172/1822198

Julia's value proposition for science

- Designed for "scientific computing" (Fortran-like) and "data science" (Python-like) with **performant kernel code via LLVM compilation**
- Lightweight **interoperability with existing Fortran and C libraries**
- Julia is a **unifying workflow language** with a **coordinated ecosystem**

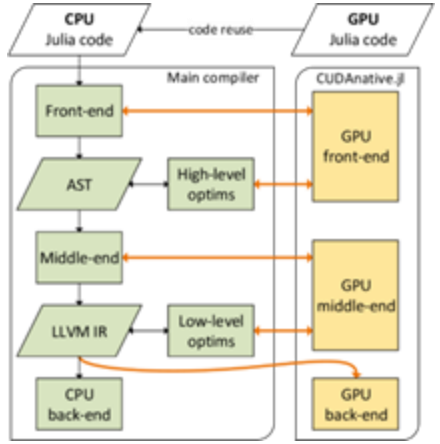


"Julia **does not** replace Python, but the costly workflow process around **Fortran+Python+X**, **C+X**, **Python+X** or **Fortran+X** (e.g. GPUs, simulation + data analysis)"
where X = { conda, pip, pybind11, Cython, Python, C, Fortran, C++, OpenMP, OpenACC, CUDA, HIP, CMake, numpy, scipy, Matplotlib, Jupyter, …**}**



https://pde-on-gpu.vaw.ethz.ch/lecture7

LLVM

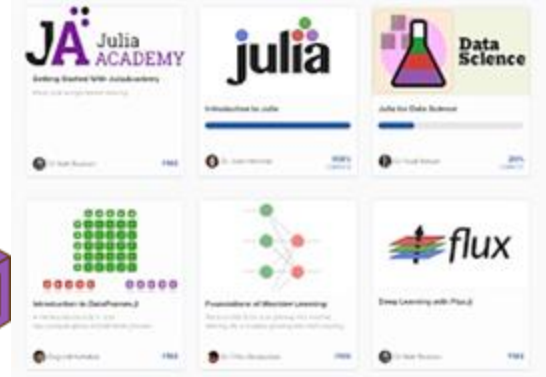


https://developer.nvidia.com/blog/gpu-computing-julia-programming-language/

https://juliadatascience.io/

Rich data science ecosystem

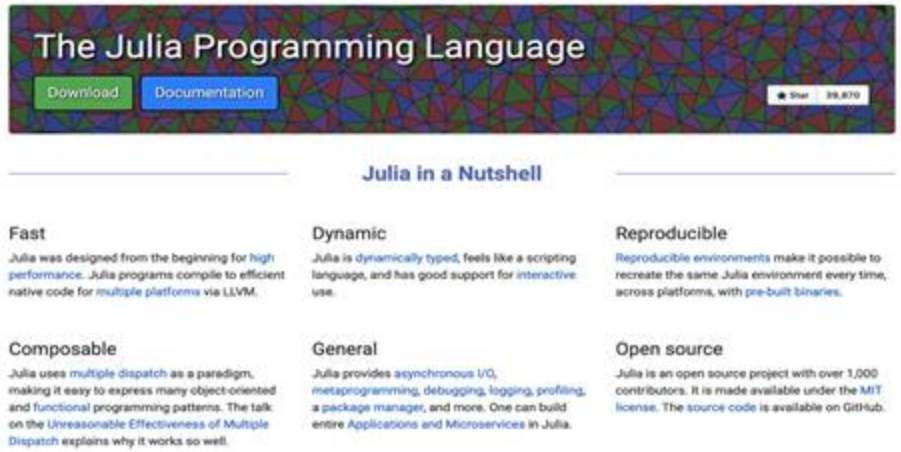
Pkg.jl

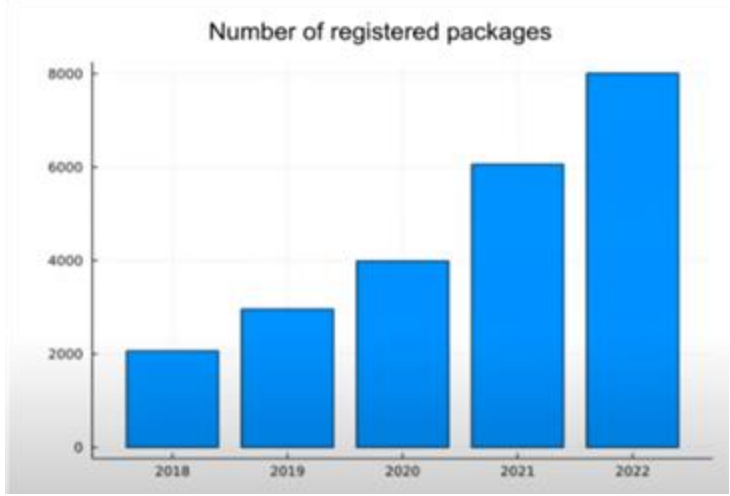


https://quantumzeitgeist.com/learning-the-julia-programming-language-for-free/

Julia Brief Walkthrough

- History: started at MIT in the early 2010s (predates Python Numba) https://julialang.org/blog/2022/02/10years/
- **JuliaHub (formerly Julia Computing)** and **MIT** are major contributors: https://info.juliahub.com/case-studies
- First stable release v1.0 in 2018, v1.11 as of 2025 https://julialang.org/
- Open-source GitHub-hosted packages and ecosystem with MIT permissive license: https://github.com/JuliaLang/julia
- Community: annual JuliaCon summer conference: https://juliacon.org/2025/

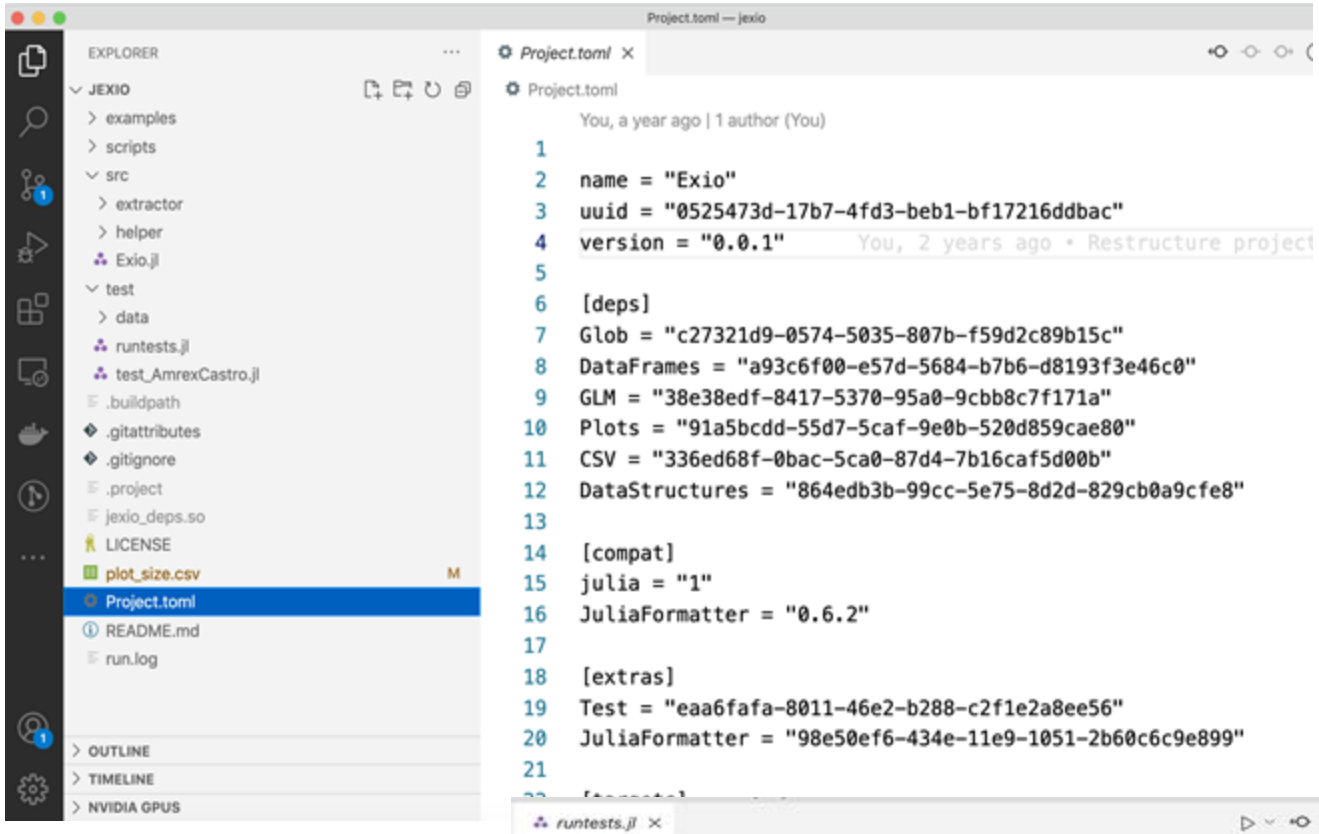




95% of Julia packages in the registry had some form of CI (youtube.com/watch?v=9YWwiFbaRx8)

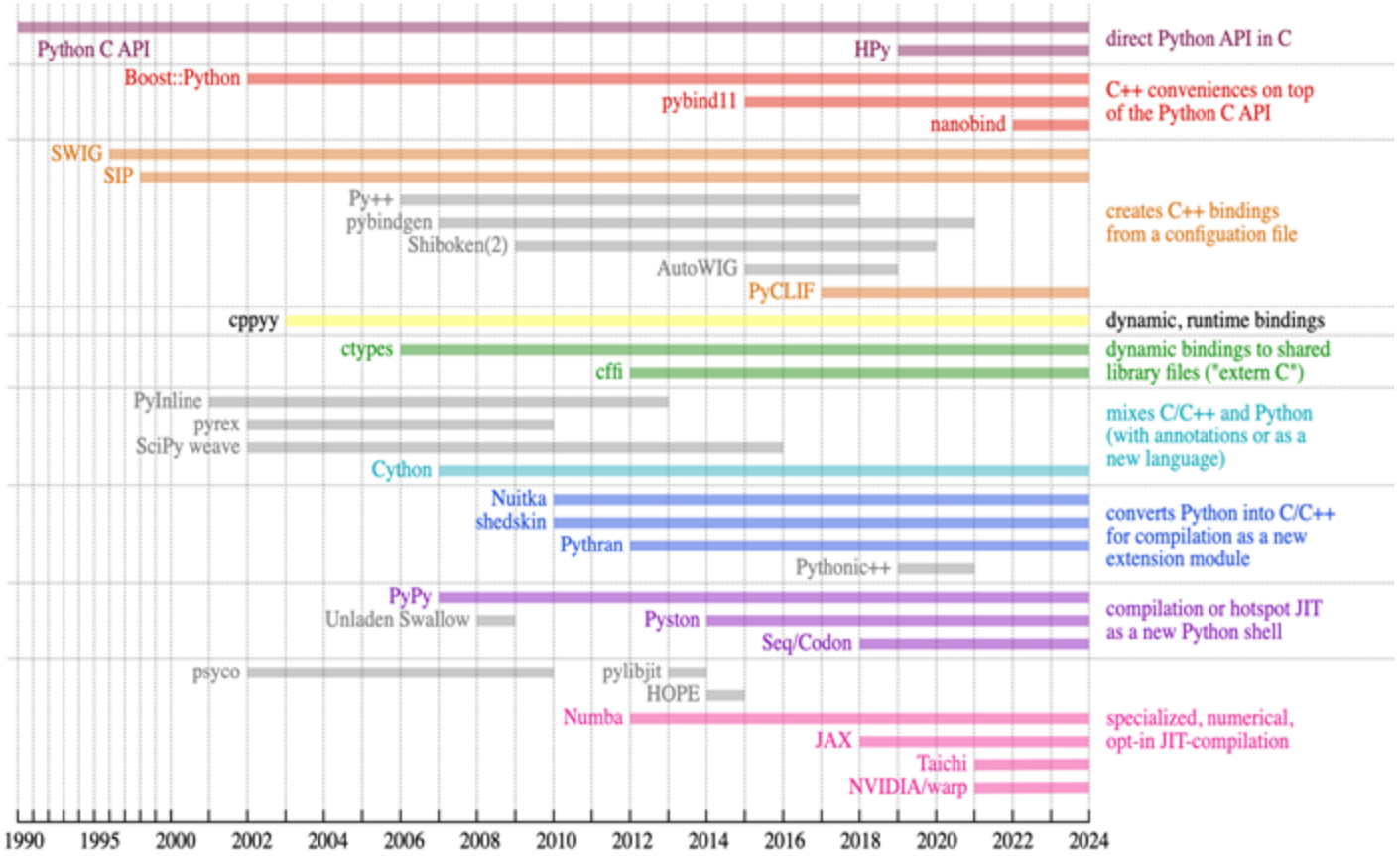
Julia Brief Walkthrough

- Reproducibility is in the core of the language:
  - Interactive: Jupyter, Pluto.jl
  - Packaging Pkg.jl
  - Environment Project.toml
  - Testing Test.jl
- Just-in-time or Ahead-of-time compilation with PackageCompiler.jl (juliac is WIP)
- Powerful metaprogramming for code instrumentation: @profile, @time, @testset, @test, @code_llvm, @code_native, @inbounds,
- Interoperability is key: @ccall, @cxx, PyCall, CxxWrap.jl



When Python is not enough…and it's a fragmented world

Making Python "faster"



https://raw.githubusercontent.com/jpivarski-talks/2023-05-01-hsf-india-tutorial/main/img/history-of-bindings-2.svg

https://pyreadiness.org/3.13/

Python 3.13 Readiness

Python 3.13 support graph for the 360 most popular Python packages!

What is this about?

Python 3.13 is a currently supported version of Python. This site shows Python 3.13 support for the 360 most downloaded packages on PyPI:

1. 161 green packages (44.7%) support Python 3.13;
2. 199 uncolored packages (55.3%) don't explicitly support Python 3.13 yet.

Package 'x' is uncolored. What can I do?

Why NOT Julia??

- *Existing large investments in other languages*
- *Long-term ROI: Package support? Stable not standard*
- *Ecosystem is not mature: Tooling? (HPC)*
- *Out of scope for my application needs*
- *I'm simply more comfortable in another language*

JACC roadmap

- JACC.experimental
  - A separate JACC module to explore new ideas
- JACC Proxies
  - Compare JACC in science workloads (LULESH, XSBench, BabelStream, Hartree-Fock)
- JACC.BLAS
  - BLAS library on top of JACC
- JACC.multi
  - Support for multi-device
- JACC.auto
  - Support for auto-tuning
- Task-based – JACC.async
  - DAGGER.jl, IRIS - R&D100

JACC
JULIA FOR ACCELERATORS


Presented at SC24 WACCPD

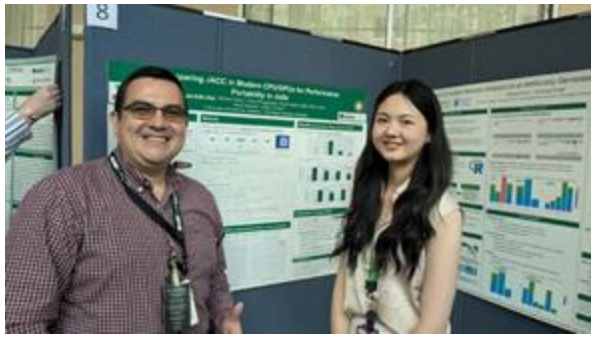
JACC: Leveraging HPC Meta-Programming and Performance Portability with the Just-in-Time and LLVM-based Julia Language

Pedro Valero-Lara, William F. Godoy, Het Mankad, Keita Teranishi, and Jeffrey S. Vetter
Oak Ridge National Laboratory
Oak Ridge, TN, USA
{valerolarap},{godoywf},{mankadhy},
{teranishik},{vetter}@ornl.gov

Johannes Blaschke
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
jpblaschke@lbl.gov

Michel Schanen
Argonne National Laboratory
Lemont, IL, USA
mschanen@anl.gov

Best ORNL CS intern poster by Kelly Tang



JACC: SC24 Best Poster Finalist (6/120)

SC24 AI4Science using Julia

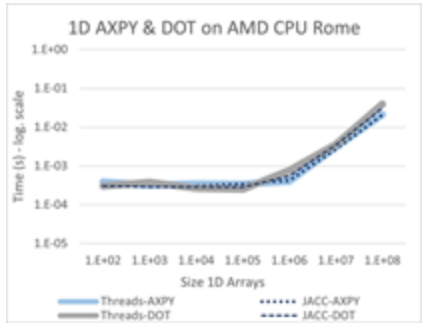
ChatBLAS: The First AI-Generated and Portable BLAS Library

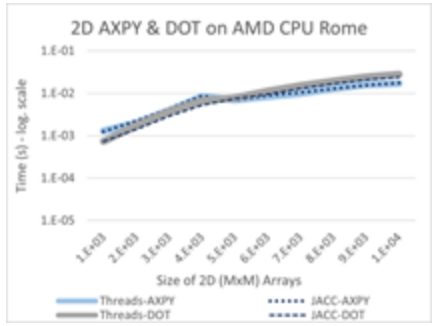
SC24: Julia for HPC 1st Tutorial and 3rd BoF w/ MIT and LBNL

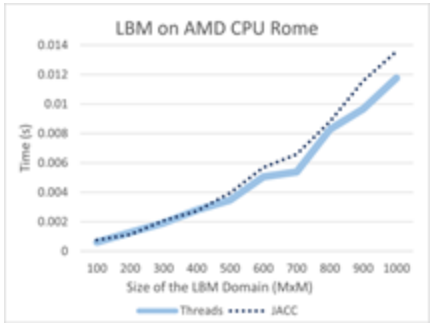
https://github.com/JuliaORNL/JACC.jl

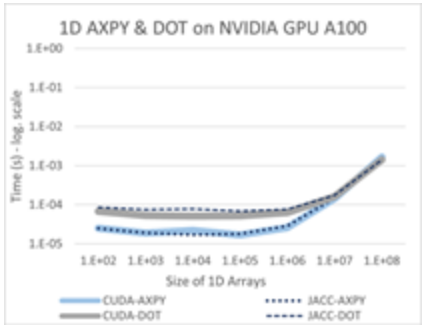
OAK RIDGE National Laboratory

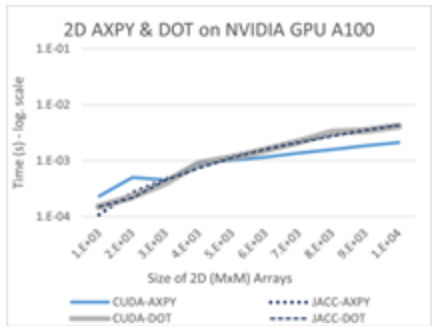
OK, but this is HPC, What about performance??

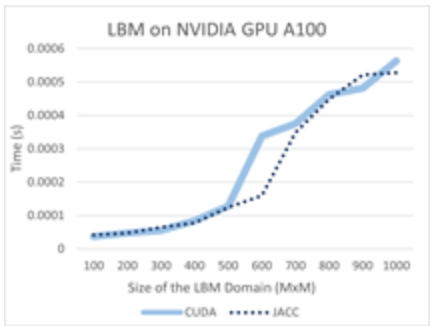


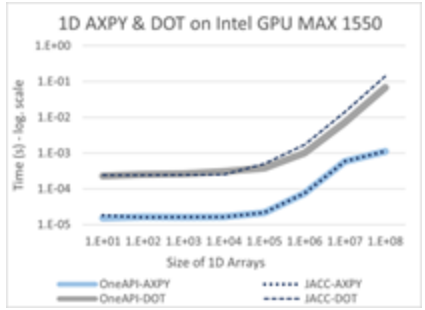


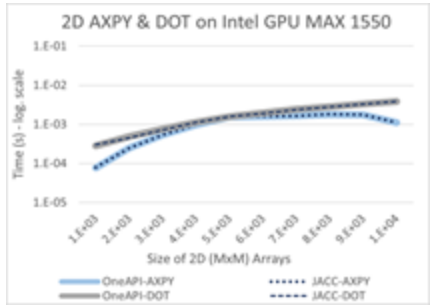


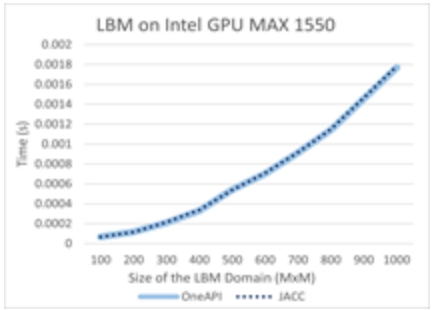


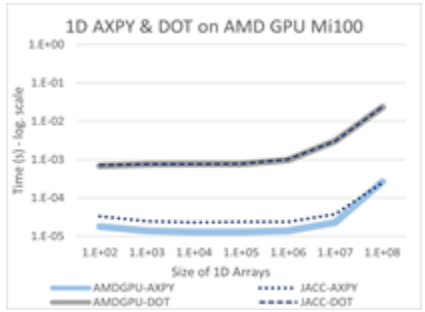


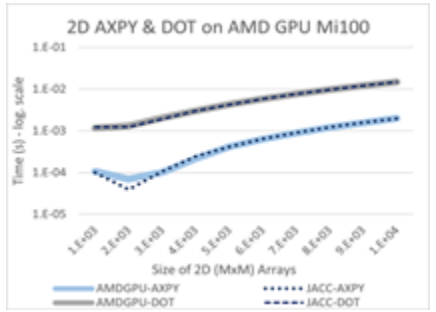


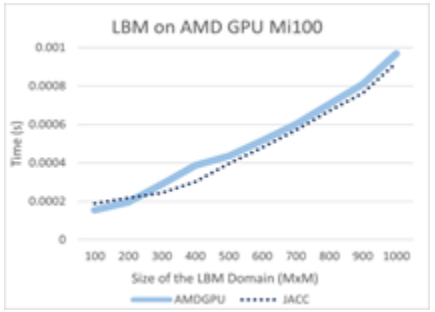












Julia as a unifying end-to-end workflow language on the Frontier exascale system. SC WORKS 2023

Evaluating performance and portability of high-level programming models: Julia, Python/Numba, and Kokkos on exascale nodes. IPDPS HIPS 2023

SC24 WACCPD

JACC: Leveraging HPC Meta-Programming and Performance Portability with the Just-in-Time and LLVM-based Julia Language

Pedro Valero-Lara, William F. Godoy, Het Mankad, Keita Teranishi, and Jeffrey S. Vetter
Oak Ridge National Laboratory
Oak Ridge, TN, USA
{valerolarap},{godoywf},{mankadhy},
{teranishik},{vetter}@ornl.gov

Johannes Blaschke
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
jpblaschke@lbl.gov

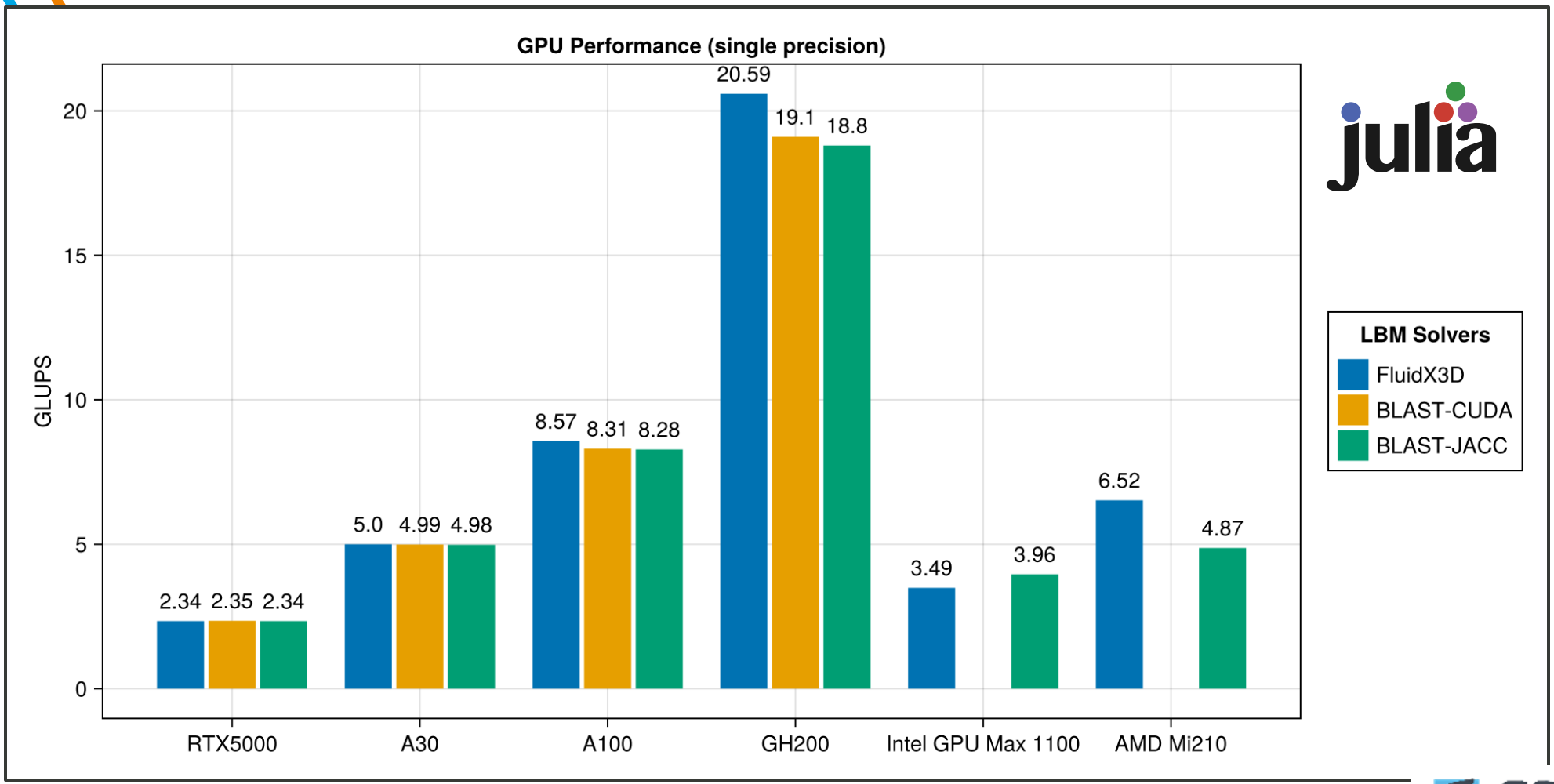
Michel Schanen
Argonne National Laboratory
Lemont, IL, USA
mschanen@anl.gov

JACC
JULIA FOR ACCELERATORS

- Lattice: D3Q19
- Mesh: 256x256x256
- Streaming algorithm: Esoteric Pull
- Test case: Taylor Green Vortex

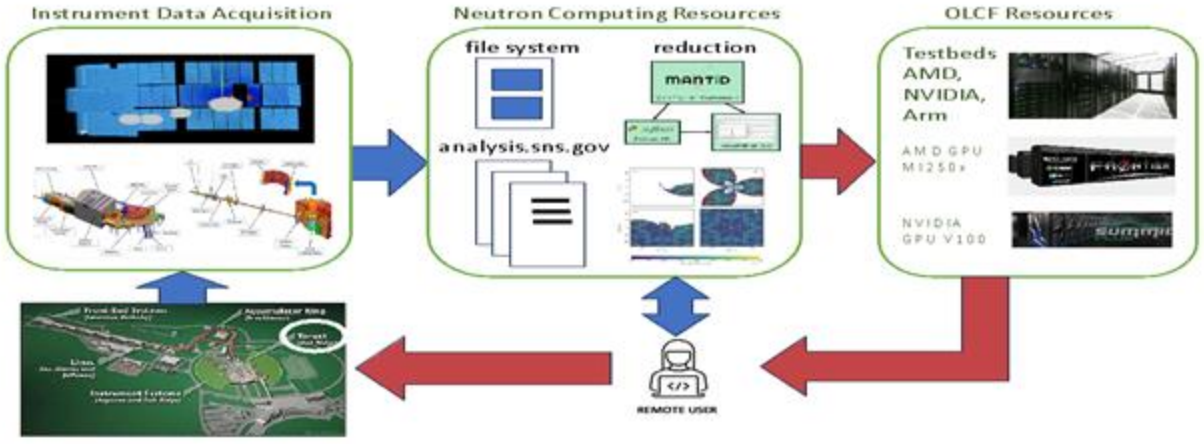
LBM Prototype in Julia

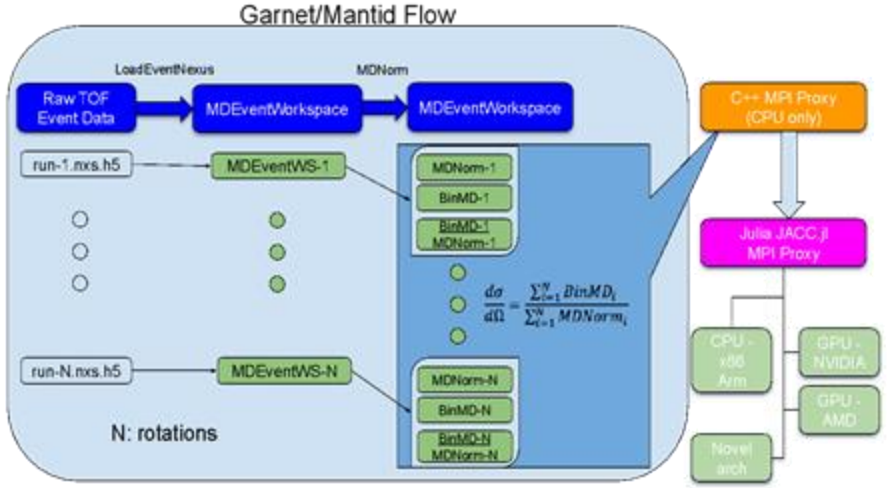
Boltzmann Lattice Advanced Simulation Tool

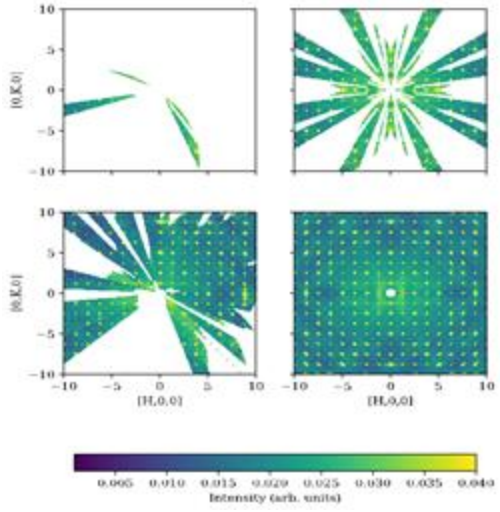


Ongoing JACC efforts: facilities

- Integrated Research Infrastructure: provide an **accessible** performance portable ecosystem
- CPU only workflows -> CPU/GPU on HPC systems







Best Paper at SC24 XLOOP

Integrating ORNL's HPC and Neutron Facilities with a Performance-Portable CPU/GPU Ecosystem

Steven E. Hahn, Philip W. Fackler, William F. Godoy, Ketan Maheshwari, Zachary Morgan, Andrei T. Savici, Christina M. Hoffmann, Pedro Valero-Lara, Jeffrey S. Vetter, and Rafael Ferreira da Silva
Oak Ridge National Laboratory, Oak Ridge, TN, USA
{hahnse,facklerpw,godoywf,km0,morganzj,saviciat,choffmann,valerolarap,vetter,silvarf}@ornl.gov

```
function binEvents!(h::Hist3, events::AbstractArray,
    transforms::Array1{SquareMatrix3c})
  JACC.parallel_for(
    (length(transforms), size(events, 2)),
    (n, i, t) -> begin
      @inbounds begin
        op = t.transforms[n]
        v = op * C3[t.events[6, i], t.events[7, i],
            t.events[8, i]]
        atomic_push!(t.h, v[1], v[2], v[3],
        t.events[1, i])
      end
    end,
    (h = h, events, transforms),
  )
end
```

Listing 3. **MiniVATES.jl BinMD CPU/GPU implementation using JACC.jl.**

Where to get started?

- Pick a gentle tutorial: https://techytok.com/from-zero-to-julia/
- https://github.com/ornl-training/julia-basics (training by WF Godoy & Philip Fackler) OLCF Tutorial: https://juliaornl.github.io/TutorialJuliaHPC/applications/GrayScott/01-Solver.html
- Use VS Code as the official IDE + debugger
- JuliaCon talks are available on YouTube
- https://discourse.julialang.org/ Stackoverflow might be outdated, https://julialang.slack.com/
- Julia docs and standard library: https://docs.julialang.org/en/v1/

- Learn: Project.toml, Testing.jl @testset @test, Pluto.jl , CUDA.jl/AMDGPU.jl , JACC.jl, KernelAbstractions.jl, LinearAlgebra.jl , Makie.jl , Plots.jl and Flux.jl (AI/ML), how to build a sysimage with PackageCompiler.jl
- **Pick problems you care about! Let us know if you're interested in a hackathon.**
- Patience and community reliance: learning a language is a big investment.

https://williamfgc.github.io/programming/scientific-computing/2020/07/22/first-project-julia-language.html

S4PST: Stewardship of Programming Systems and Tools (2024-2029)

PI: Keita Teranishi (ORNL), Co-PIs: Pedro Valero-Lara, William F Godoy

8 National Laboratories:

- Oak Ridge National Laboratory
- Argonne National Laboratory
- Lawrence Livermore National Laboratory
- Lawrence Berkeley National Laboratory
- Sandia National Laboratories
- Brookhaven National Laboratory
- Los Alamos National Laboratory
- SLAC National Accelerator Laboratory

University Partners:

- University of Delaware
- Massachusetts Institute of Technology

Collaborations:

- Louisiana State University
- Pacific Northwest National Laboratory
- Carnegie Mellon University
- University of Tennessee, Knoxville
- Stanford University
- Other 6 NSSGT projects

OAK RIDGE National Laboratory

20

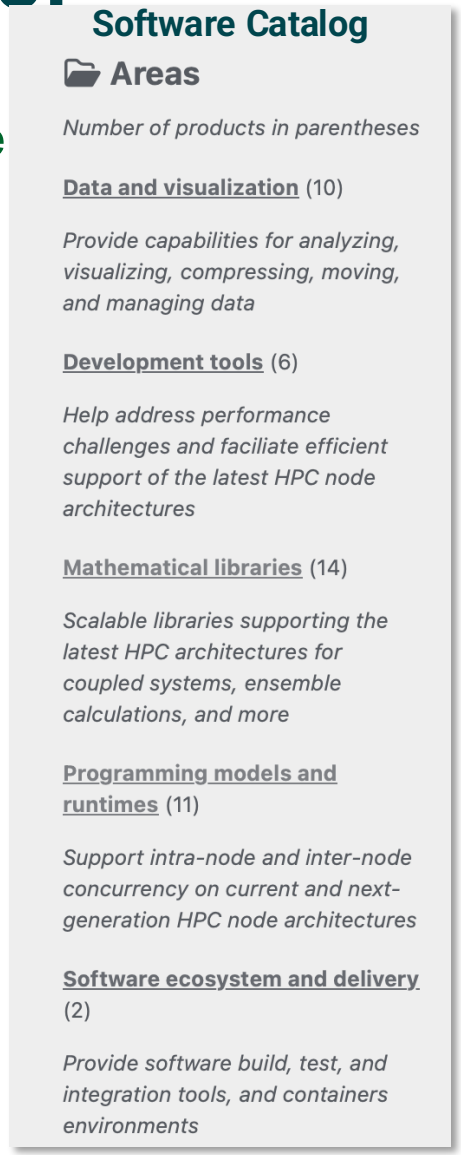
Consortium for the Advancement of Scientific Software (CASS)

CASS and its member organizations work with our software product teams to improve the quality, sustainability, and interoperability of the software products in our ecosystem

The CASS software catalog covers a range of freely available libraries supporting leading-edge computational science and engineering research on high-performance computers. Most products available via Spack in the E4S distribution.

Engage with us:

- Learn about the impacts of CASS software
- Join our announcement mailing list
- Participate in our working groups
- Reach out to a member organization responsible for specific areas of the software ecosystem of interest
- Become a CASS member. We welcome projects and organizations with similar scientific software stewardship missions



PESO

https://pesoproject.org

Scientific software ecosystem benefits (technical and community)

100,000+
Lines of code replaced with high-quality libraries and tools

10,000+
Community members via ecosystem collaborations

1,000+
Code teams share ecosystem costs and benefits

100+
Speedup using advanced devices like GPUs

10+
Reduction in build times via Spack build caches

1
Source code base for all computing systems

OAK RIDGE National Laboratory

Current CASS members: CORSA, FASTMath, PESO, RAPIDS, S4PST, STEP, COLABS, SWAS

Current funding: DOE/ASCR NGSST and SciDAC programs

https://cass.community

Summary

- We explore the value proposition of Julia for our needs: modern science language on top of LLVM

- **Q: What's Julia's place in the DOE?**
  - **Heterogenous targets, HPC, Quantum, Energy Efficiency, AI, LLMs?**

ORNL:
- **Philip Fackler (here today)**
- Pedro Valero-Lara
- Steven E Hahn
- Keita Teranishi
- Jeffrey S Vetter
- Rafael Ferreira da Silva
- Steven E. Hahn
- Corinna Thomas

DOE and External Collaborations:
- Suzanne Parete-Koon (NCCS)
- Helen He (NERSC)
- Johannes Blaschke (NERSC)
- Mose Giordano (UCL, UK)
- Rabab Alomairy, Julian Samaroo, Alan Edelman (MIT)
- Patrick Diehl, Kipton Barros (ANL)
- ORNL Workshop presenters

OAK RIDGE National Laboratory

Acknowledgements

This research used resources of the Oak Ridge Leadership Computing Facility and the Experimental Computing Laboratory at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Sponsors:
ASCR Competitive Portfolios: MAGMA/Fairbanks project
ASCR NGSST PESO and S4PST projects
ASCR Facilities: OLCF
Past: The ASCR Bluestone Project and ECP PROTEAS-TUNE

Julia Community

JACC JULIA FOR ACCELERATORS

CASS

If you are ready to explore Julia, there is a community https://scientificcoder.com/my-target-audience

Thanks to the audience!

We will be back at 3:20 pm EDT