JULIA FOR HPC ON OLCF SYSTEMS

William F Godoy, Pedro Valero-Lara, Philip Fackler

Computer Science and Mathematics Division

Advanced Computing Systems Research Section

Prepared for: SEPTEMBER 2022 OLCF USER CONFERENCE CALL 09/27/2022

Contents

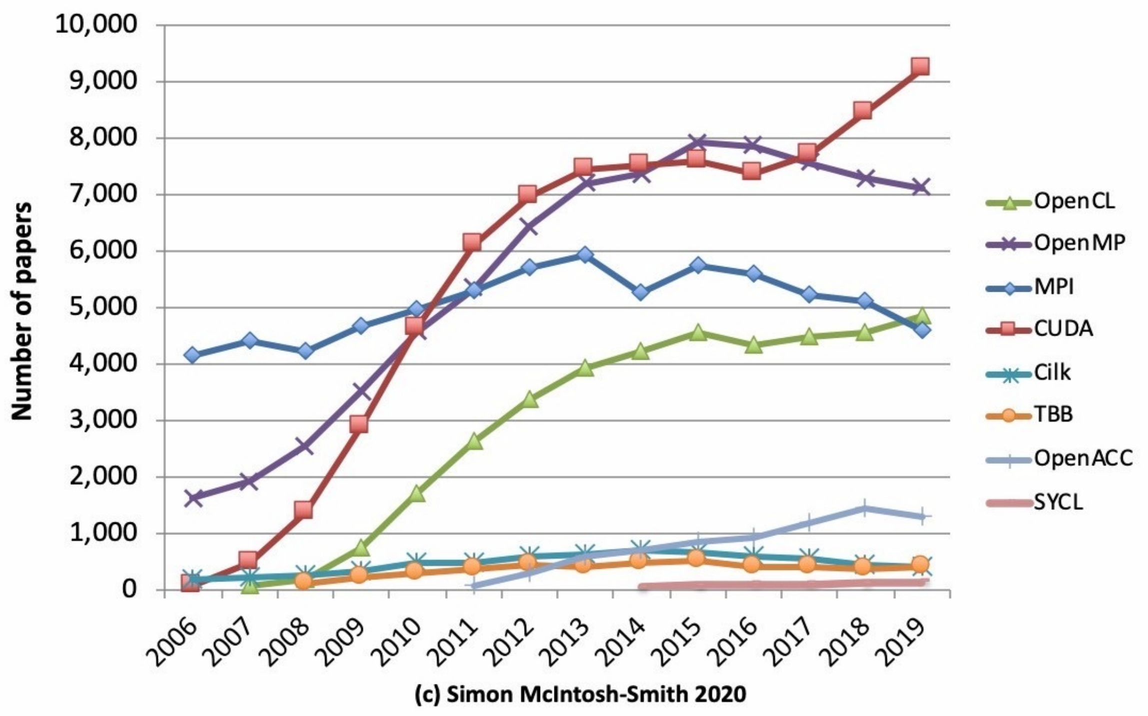
- Julia’s value proposition for HPC: LLVM + Coordinated Ecosystem
- Community Efforts in HPC
- Running on OLCF System, preliminary results, opportunities
- Resources: where to get started?
- Final thoughts and acknowledgments

Contents

- **Julia's value proposition for HPC: LLVM + Coordinated Ecosystem**
- Community Efforts in HPC
- Running on OLCF System, preliminary results, opportunities
- Resources: where to get started?
- Final thoughts and acknowledgments

Landscape of computing: The Tower of Babel

- Scientific software:
  - 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, NVIDIA/AMD GPU, Intel's Xeon, KNL, Sapphire Rapids, AVX-512
- AI/ML + 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/

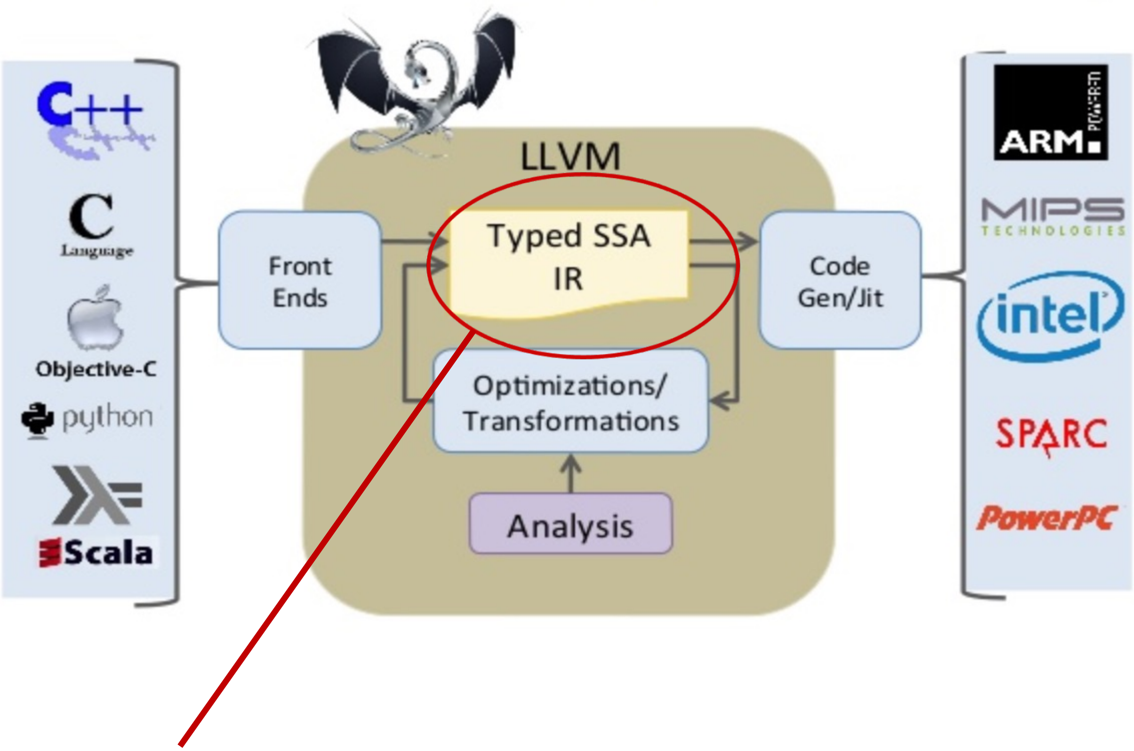
LLVM: a game changer https://llvm.org/

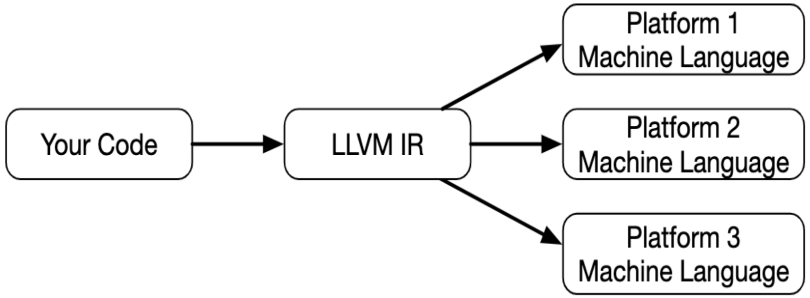
http://www.aosabook.org/en/llvm.html

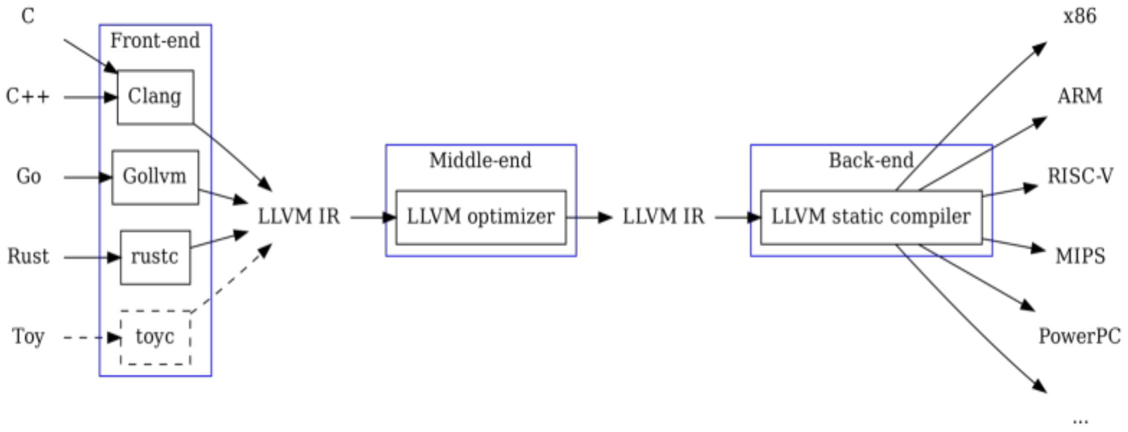
C. Lattner and V. Adve, "LLVM: a compilation framework for lifelong program analysis & transformation," *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, 2004, pp. 75-86, https://doi.org/10.1109/CGO.2004.1281665 .

LLVM Compiler Infrastructure

[Lattner et al.]







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

OpenACC Directives for Accelerators

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

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

OpenMP

https://openmp.llvm.org

NVIDIA

NVIDIA CUDA 4.1 Compiler Now Built on LLVM

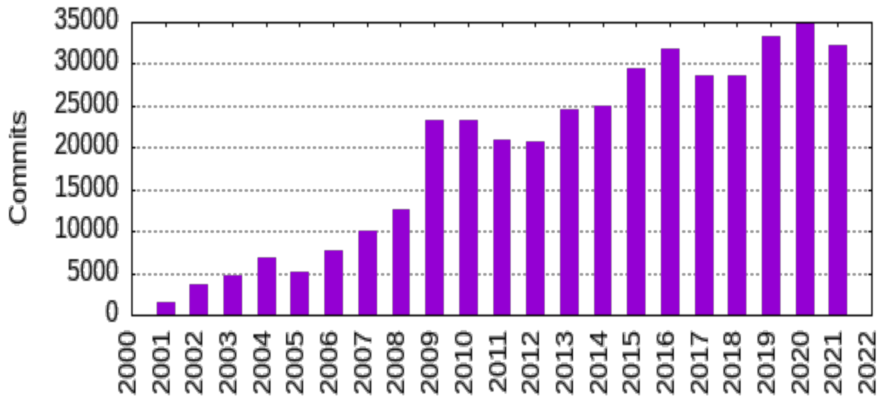
By Chris Lattner
Dec 19, 2011

NVIDIA CUDA

LLVM Commits

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.



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,

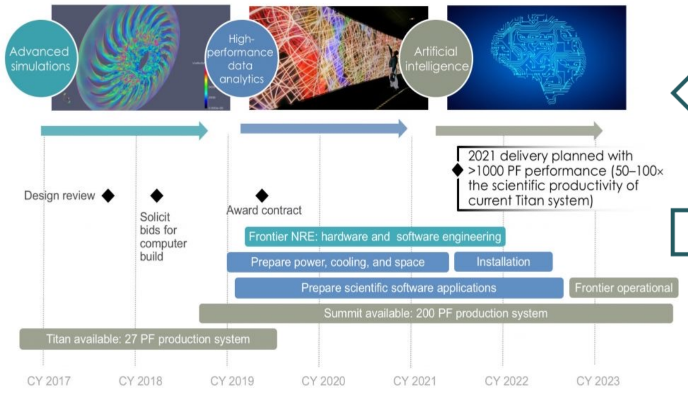
AMD ROCm

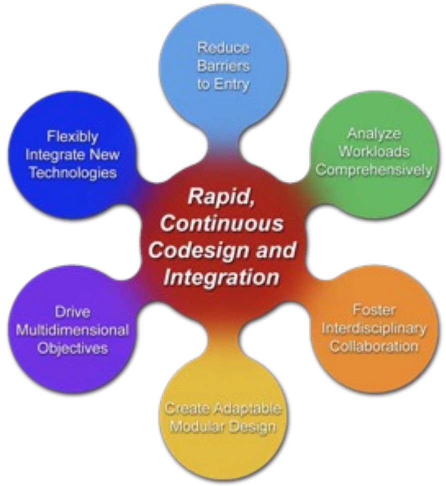
Oak Ridge National Laboratory

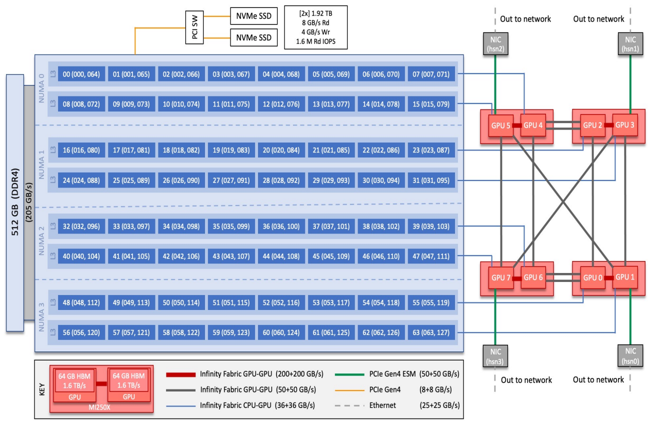
Rethink how we do Computing

- Scientific programming is HARD (specially on our Leadership Computing Facilities, LCFs)
- Software is our “specialized science equipment” for science
- There is still a lot of plumbing to be done
- Programming productivity is always a challenge
- Barrier to entry from idea to portable performance
- AI/ML+HPC is a multidisciplinary co-design challenge
- How to leverage ECP legacy?

“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)



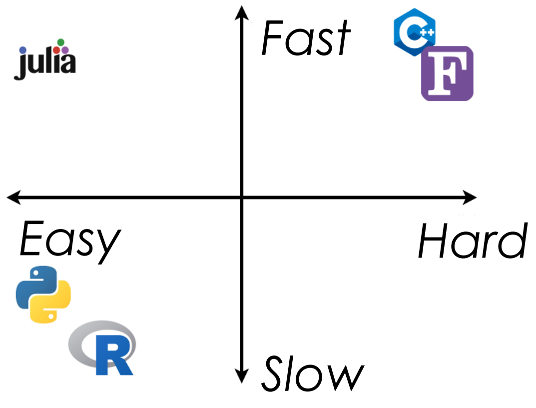




Key question: “What novel approaches to software design and implementation can be developed to provide performance portability for applications across radically diverse computing architectures?” from Reimagining Codesign for Advanced Scientific Computing: Unlocking Transformational Opportunities for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

Julia's value proposition for HPC

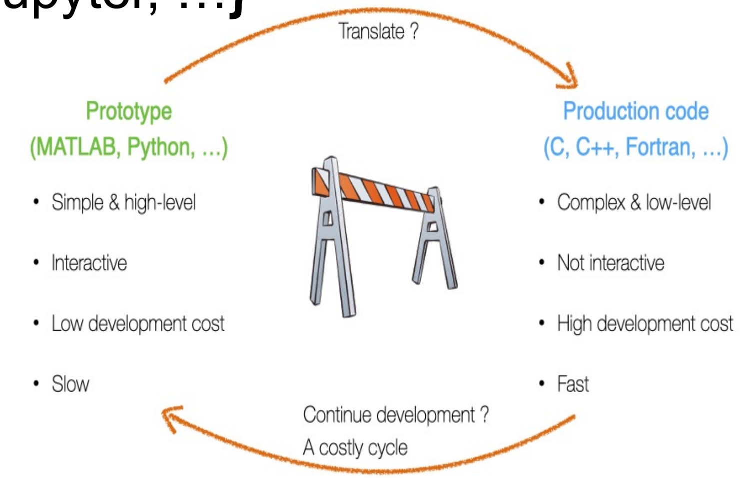
- Designed for "scientific computing" (Fortran-like) and "data science" (Python-like) with **performant kernel code via LLVM compilation**
- Lightweight **interoperability with existing Fortran and C libraries**
- Julia is a **unifying workflow language** with a **coordinated ecosystem**



https://juliadatascience.io/

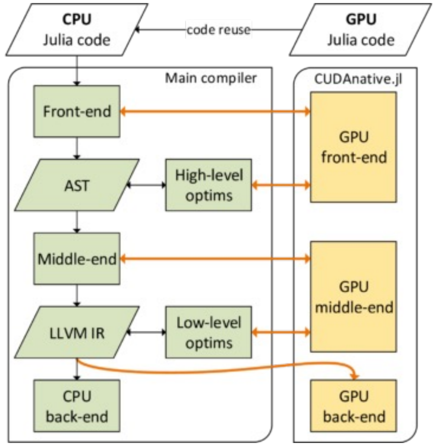
"Julia **does not** replace Python, but the costly workflow process around **Fortran+Python+X**, **C+X**, **Python+X** or **Fortran+X** (e.g. GPUs)"

where X = { conda, pip, pybind11, cython, Python, C, Fortran, C++, OpenMP, OpenACC, CUDA, HIP, CMake, numpy, scipy, matplotlib, Jupyter, …**}**



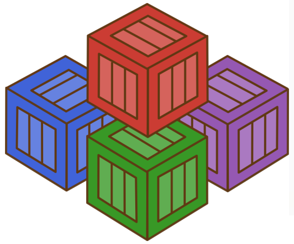
https://pde-on-gpu.vaw.ethz.ch/lecture7

LLVM

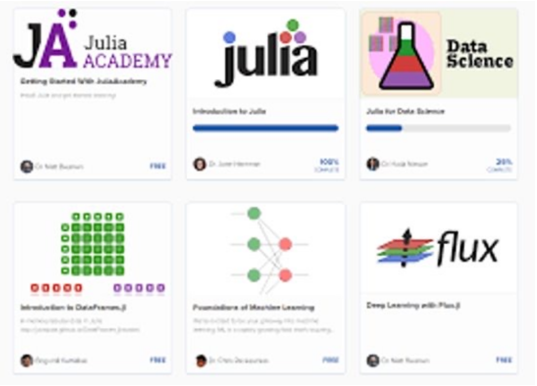


https://developer.nvidia.com/blog/gpu-computing-julia-programming-language/

Pkg.jl



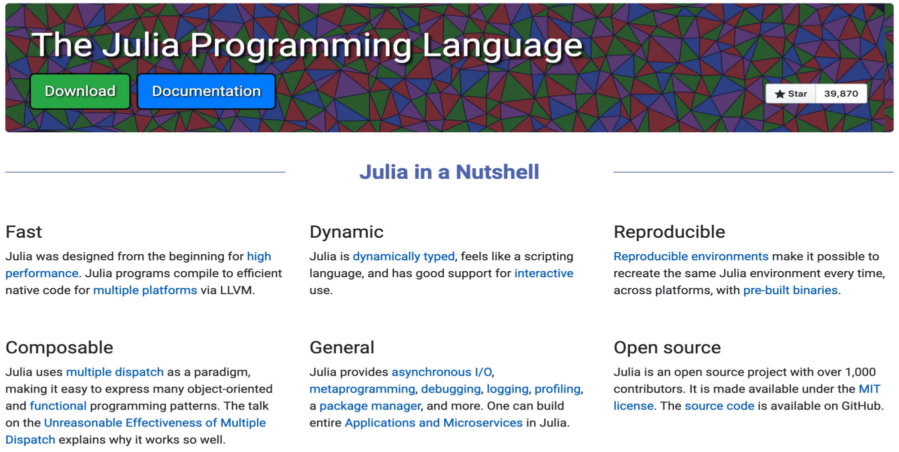
Rich data science ecosystem



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

Julia Brief Walkthrough

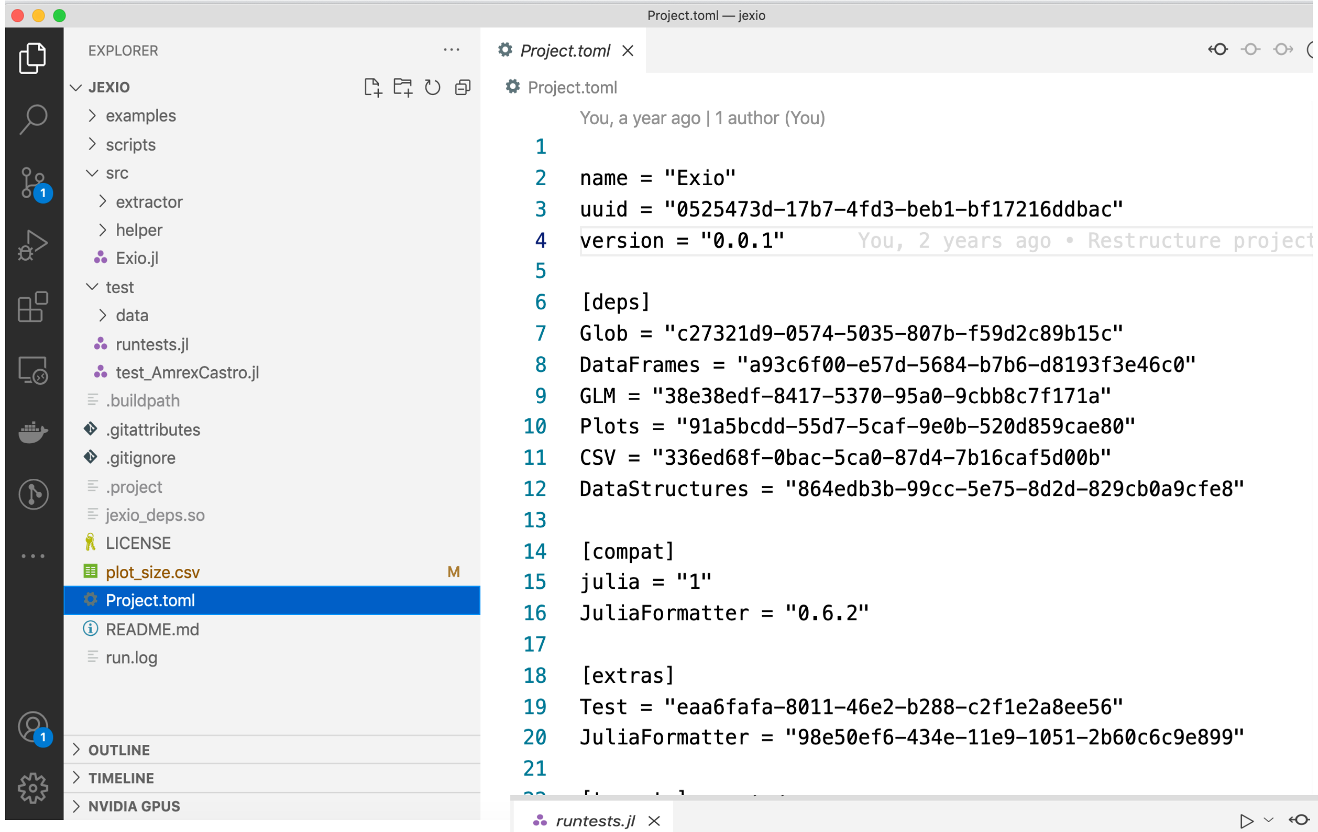
- History: started at MIT in the early 2010s (predates Python Numba) https://julialang.org/blog/2022/02/10years/
- Julia Computing is a major contributor: https://juliacomputing.com/case-studies
- First stable release v1.0 in 2018, v1.8 as of 2022 https://julialang.org/
- Open-source GitHub-hosted packages and ecosystem with MIT permissive license: https://github.com/JuliaLang/julia
- Community: annual JuliaCon conference (next week): https://juliacon.org/2022/ https://live.juliacon.org/agenda/2022-07-19

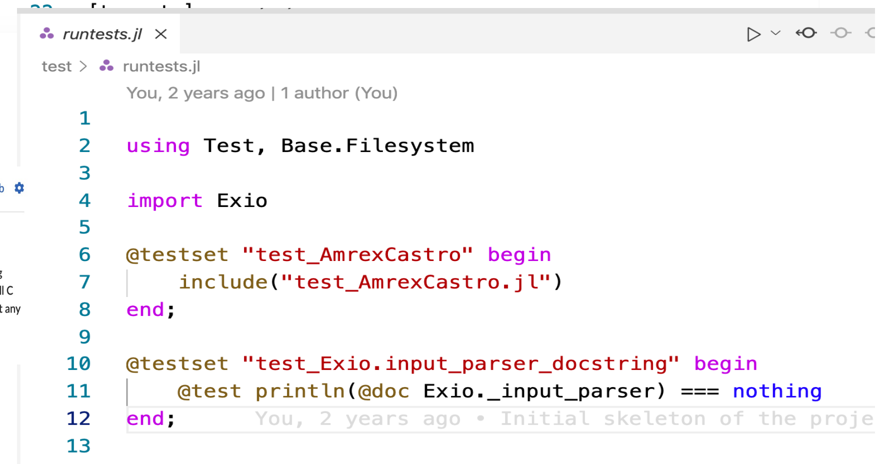


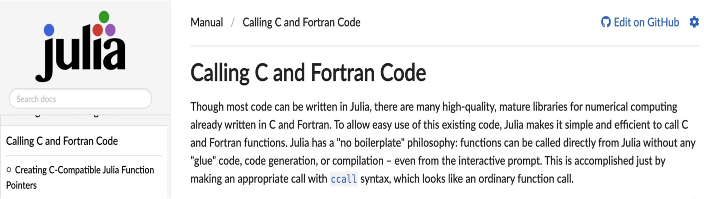
Oak Ridge National Laboratory

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
- ❑ Powerful metaprogramming for code instrumentation: @profile, @time, @testset, @test, @code_llvm, @code_native, @inbounds,
- ❑ Interoperability is key: @ccall, @cxx, PyCall, CxxWrap.jl







https://github.com/ornl-training/julia-basics/tree/main/notebooks/MPI-Heat-Transfer-2D

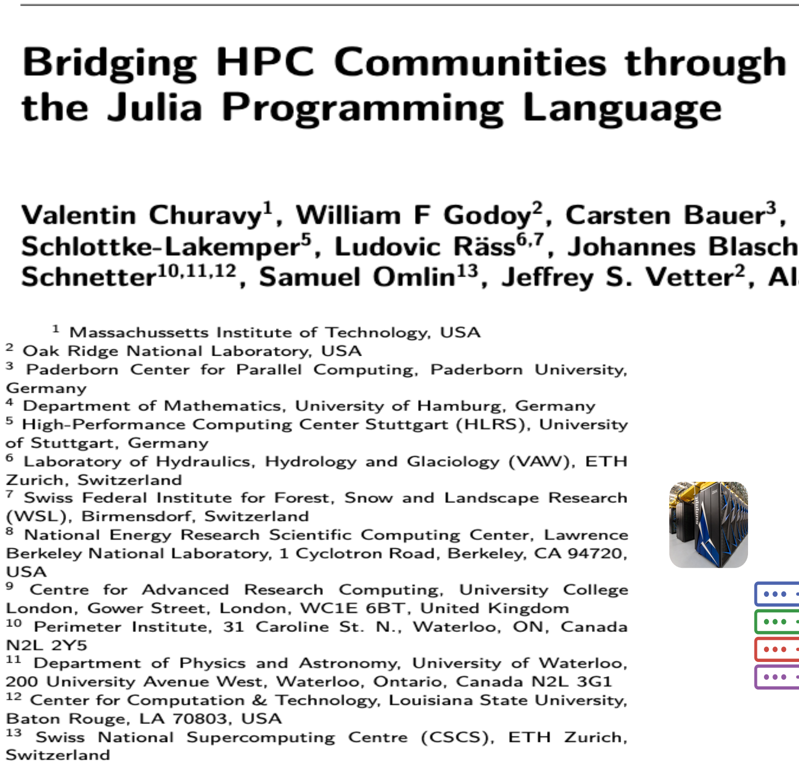
	Fortran	C	C++	Python	Julia
Package manager	fpm, Too many	Too many	Too many	3rd party: pip, conda	Pkg.jl
Environment Reproducibility	?	?	?	3rd party: pyproject.toml	Project.toml
Metaprogramming	fpp	macros #	Templates, macros #	decorators @	macros @
Memory management	Manual	Manual	RAII, Manual	Garbage Collected	Garbage Collected
Performance	Compiled	Compiled	Compiled	3rd party: numba (LLVM), numpy, pybind11, Cython	JIT and/or AOT compiled: PackageCompiler.jl
Testing	3rd party	3rd party	3rd party	3rd party: pytest, unittest,	@testset
TIOBE index (Sep 2022)	15 (17)	2	4	1	21 (27)

Contents

- Julia’s value proposition for HPC: LLVM + Coordinated Ecosystem
- **Community Efforts in HPC**
- Running on OLCF System, preliminary results, opportunities
- Resources: where to get started?
- Final thoughts and acknowledgments

CCSD efforts

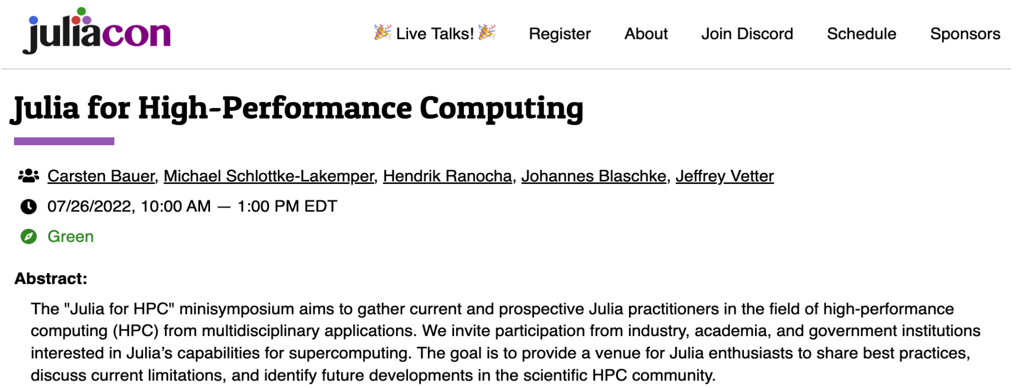
- Projects:
  - **ECP PROTEAS-TUNE:** research performance on Exascale system of different programming models including Python Numba and Julia (W Godoy, J Vetter). SRP-HPC mentor
  - **ECP Proxy Apps:** evaluating Julia as a proxy to understand parallel I/O characteristics (W Godoy, P Fackler, G Watson) RIOPA.jl presented at JuliaCon 2022
- External Engagements with Julia HPC
  - Monthly meetings with stakeholders
  - JuliaCon 2022 HPC minisymposium
  - Julia HPC Position paper in the works
  - ECP BoF Session on Rapid Prototyping for HPC using Julia, Python Numba, Flang
  - SC22 BoF "Julia for HPC"
- Internal Engagements
  - Tutorials at ORNL Software and Data Expo
  - Julia Workshop for ORNL Science


Bridging HPC Communities through the Julia Programming Language

Journal Title
XX(X):1–18
©The Author(s) 2022
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Valentin Churavy[1], William F Godoy[2], Carsten Bauer[3], Hendrik Ranocha[4], Michael Schlottke-Lakemper[5], Ludovic Räss[6,7], Johannes Blaschke[8], Mosè Giordano[9], Erik Schnetter[10,11,12], Samuel Omlin[13], Jeffrey S. Vetter[2], Alan Edelman[1]

[1] Massachussetts Institute of Technology, USA
[2] Oak Ridge National Laboratory, USA
[3] Paderborn Center for Parallel Computing, Paderborn University, Germany
[4] Department of Mathematics, University of Hamburg, Germany
[5] High-Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Germany
[6] Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Switzerland
[7] Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland
[8] National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA
[9] Centre for Advanced Research Computing, University College London, Gower Street, London, WC1E 6BT, United Kingdom
[10] Perimeter Institute, 31 Caroline St. N., Waterloo, ON, Canada N2L 2Y5
[11] Department of Physics and Astronomy, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1
[12] Center for Computation & Technology, Louisiana State University, Baton Rouge, LA 70803, USA
[13] Swiss National Supercomputing Centre (CSCS), ETH Zurich, Switzerland



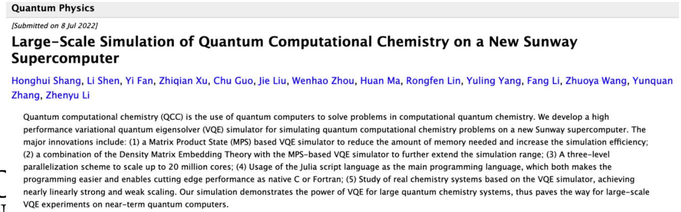
Julia for High-Performance Computing

Carsten Bauer, Michael Schlottke-Lakemper, Hendrik Ranocha, Johannes Blaschke, Jeffrey Vetter
07/26/2022, 10:00 AM — 1:00 PM EDT
Green

Abstract:
The "Julia for HPC" minisymposium aims to gather current and prospective Julia practitioners in the field of high-performance computing (HPC) from multidisciplinary applications. We invite participation from industry, academia, and government institutions interested in Julia's capabilities for supercomputing. The goal is to provide a venue for Julia enthusiasts to share best practices, discuss current limitations, and identify future developments in the scientific HPC community.

Community Efforts in HPC

- Leverage HPC "backends":
  - AMDGPU.jl
  - CUDA.jl
  - KernelAbstractions.jl
  - MPI.jl
  - Threads (part of Base)
  - ADIOS2 , HDF5
- Monthly HPC Call (Valentin Churavy, MIT)
- Porting miniWeather App to Julia (Youngsung Kim, Hyun Kang, and Sarat Sreepathi, CSED)
- https://ptsolvers.github.io/GPU4GEO/software/
- https://arxiv.org/abs/2207.03711


Quantum Physics

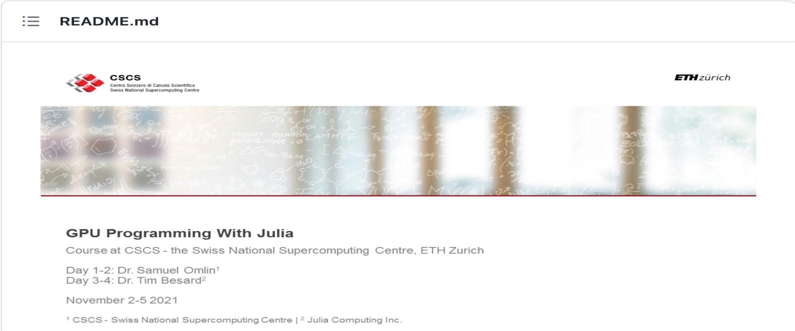
[Submitted on 8 Jul 2022]

Large-Scale Simulation of Quantum Computational Chemistry on a New Sunway Supercomputer

Honghui Shang, Li Shen, Yi Fan, Zhiqian Xu, Chu Guo, Jie Liu, Wenhao Zhou, Huan Ma, Rongfen Lin, Yuling Yang, Fang Li, Zhuoya Wang, Yunquan Zhang, Zhenyu Li

Quantum computational chemistry (QCC) is the use of quantum computers to solve problems in computational quantum chemistry. We develop a high performance variational quantum eigensolver (VQE) simulator for simulating quantum computational chemistry problems on a new Sunway supercomputer. The major innovations include: (1) a Matrix Product State (MPS) based VQE simulator to reduce the amount of memory needed and increase the simulation efficiency; (2) a combination of the Density Matrix Embedding Theory with the MPS-based VQE simulator to further extend the simulation range; (3) a three-level parallelization scheme to scale up to 20 million cores; (4) Usage of the Julia script language as the main programming language, which both makes the programming easier and enables cutting edge performance as native C or Fortran; (5) Study of real chemistry systems based on the VQE simulator, achieving nearly linearly strong and weak scaling. Our simulation demonstrates the power of VQE for large quantum chemistry systems, thus paves the way for large-scale VQE experiments on near-term quantum computers.

https://github.com/omlins/julia-gpu-course


README.md

cscs Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

ETH zürich

GPU Programming With Julia

Course at CSCS - the Swiss National Supercomputing Centre, ETH Zurich

Day 1-2: Dr. Samuel Omlin[1]
Day 3-4: Dr. Tim Besard[2]

November 2-5 2021

[1] CSCS - Swiss National Supercomputing Centre | [2] Julia Computing Inc.

https://enccs.github.io/Julia-for-HPC


Julia for High-Performance Scientific Computing

ENCCS EuroCC National Competence Centre Sweden

Search docs

» Julia for high-performance scientific computing — Edit on GitHub

Julia for high-performance scientific computing

Julia is a scientific programming language that is free and open source - see https://julialang.org/ for downloads, documentation, learning resources etc. Bridging high-level interpreted and low-level compiled languages, it offers high performance (comparable to C and Fortran) without sacrificing simplicity and programming productivity (like in Python or R).

Julia has a rich ecosystem of libraries aimed towards scientific computing and a powerful in-built package manager to install and manage their dependencies. Julia is also gaining ground in HPC as it supports both threading and distributed-memory parallelization as well as GPU computing.

https://docs.dftk.org/stable


DFTK.jl

Home — Edit on GitHub

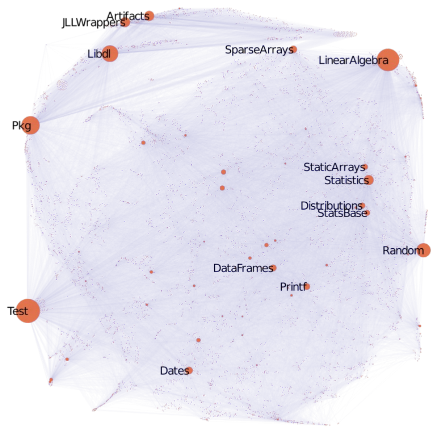
DFTK.jl: The density-functional toolkit.

The density-functional toolkit, DFTK for short, is a library of Julia routines for playing with plane-wave density-functional theory (DFT) algorithms. In its basic formulation it solves periodic Kohn-Sham equations. The unique

https://juliaastro.github.io/dev

https://github.com/JuliaParallel

Top15 most popular packages



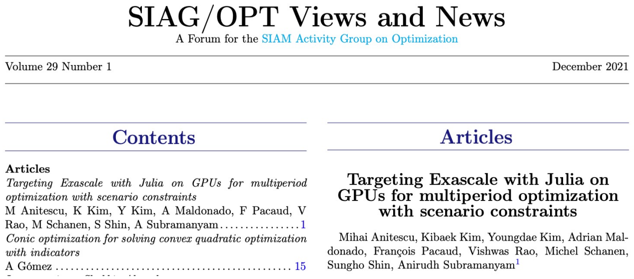
ECP ExaSDG on Summit


Research | July 06, 2022 — Print

Rapid Prototyping with Julia: From Mathematics to Fast Code

By Michel Schanen, Valentin Churavy, Youngdae Kim, and Mihai Anitescu

Software development—a dominant expenditure for scientific projects—is often limited by technical programming challenges, not mathematical insight. Here we share our experience with the Julia programming language in the context of the U.S. Department of Energy's Exascale Computing Project (ECP) as part of ExaSGD, a power grid optimization application. Julia is a free and open-source language that has the potential for C-like performance


SIAG/OPT Views and News

A Forum for the SIAM Activity Group on Optimization

Volume 29 Number 1 — December 2021

Contents

Articles

Targeting Exascale with Julia on GPUs for multiperiod optimization with scenario constraints
M Anitescu, K Kim, Y Kim, A Maldonado, F Pacaud, V Rao, M Schanen, S Shin, A Subramanyam 1
Conic optimization for solving convex quadratic optimization with indicators
A Gómez 15

Articles

Targeting Exascale with Julia on GPUs for multiperiod optimization with scenario constraints

Mihai Anitescu, Kibaek Kim, Youngdae Kim, Adrian Maldonado, François Pacaud, Vishwas Rao, Michel Schanen, Sungho Shin, Anirudh Subramanyam[1]

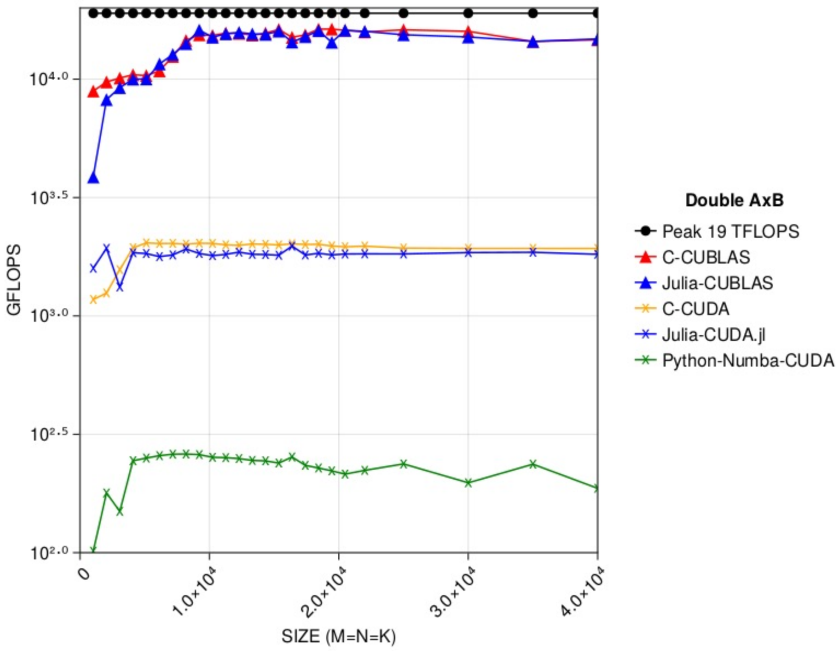
National Laboratory

Contents

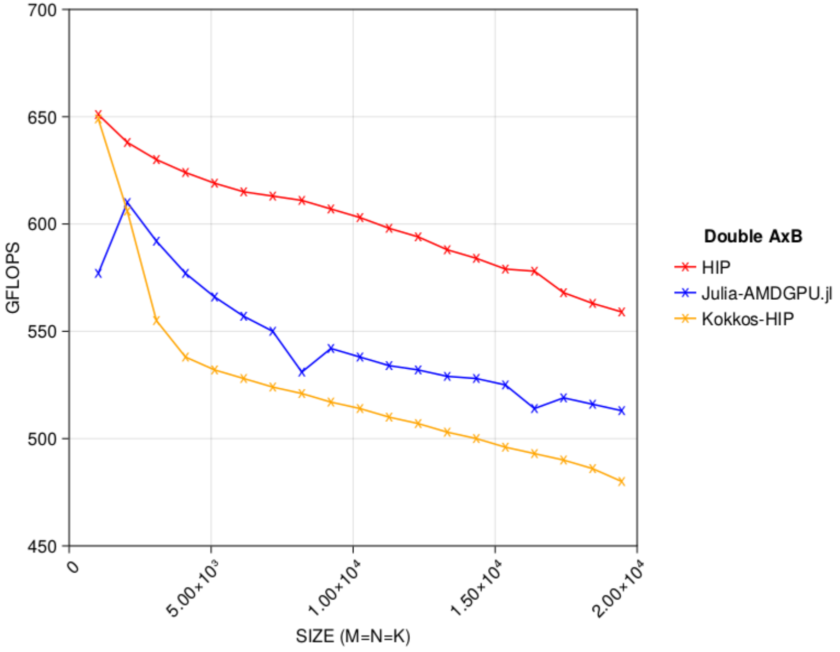
- Julia's value proposition for HPC: LLVM + Coordinated Ecosystem
- Community Efforts in HPC
- **Running on OLCF System, preliminary results, opportunities**
- Resources: where to get started?
- Final thoughts and acknowledgments

Performance results on GPU
Results for Matrix Multiplication

Wombat (ARM)
NVIDIA A100 GPU



Crusher (AMD)
MIX250X



https://github.com/williamfgc/simple-gemm/tree/main/scripts/julia

Julia and Python's Numba kernel portability, performance, and productivity on heterogeneous exascale nodes

William F. Godoy, Pedro Valero-Lara, T. Elise Dettling, Christian Trefftz, Ian Jorquera, Thomas Sheehy, Ross G. Miller Marc Gonzalez-Tallada, Jeffrey S Vetter
Oak Ridge National Laboratory
{godoywf}, {valerolarap}, {dettlingte}, {trefftzci}, {jorqueraid}, {sheehytb}, {rgmiller}, {gonzaleztal}, {vetter}@ornl.gov

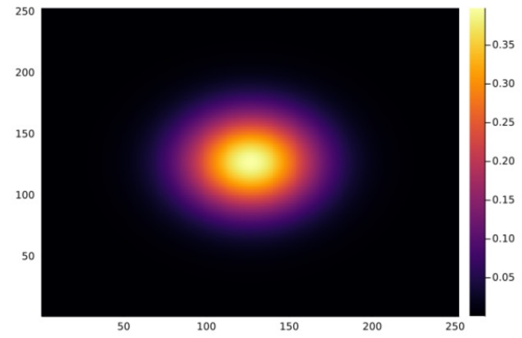
ROCm-MPI

ROCm (-aware) MPI tests on AMD GPUs on following platforms:

- Ault test system (MI50)
- LUMI-G supercomputer (MI250x)
- Crusher - Frontier's test bed (MI250x)

Multi AMD-GPU results (on LUMI-G eap)

1000 diffusion steps on 4 MI250x GPUs



https://github.com/luraess/ROCm-MPI

Interactive computing on JupyterHub with Julia

```
(v1.6) pkg> add IJulia
... (Or install IJulia any other way).
julia> using IJulia
julia> installkernel("Julia (16 threads)", env=Dict("LD_LIBRARY_PATH" => "",
"JULIA_NUM_THREADS"=>"16"))
```

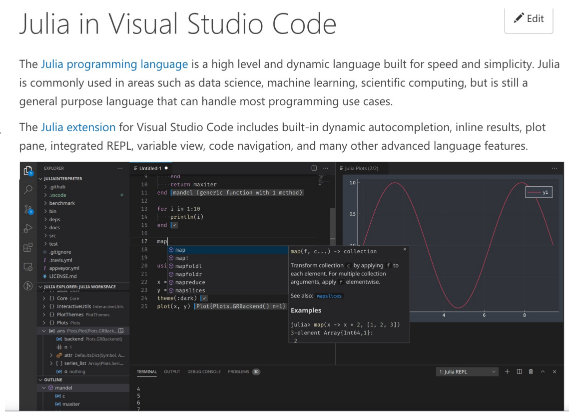
…or use Pluto.jl on a terminal

Contents

- Julia’s value proposition for HPC: LLVM + Coordinated Ecosystem
- Community Efforts in HPC
- Running on OLCF System, preliminary results, opportunities
- **Resources: where to get started?**
- Final thoughts and acknowledgments

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 Fackler)
- Use VS Code as the official IDE + debugger (not Juno)
- 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 , LinearAlgebra.jl , Makie.jl , Plots.jl and Flux.jl (AI/ML), how to build a sysimage
- **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.



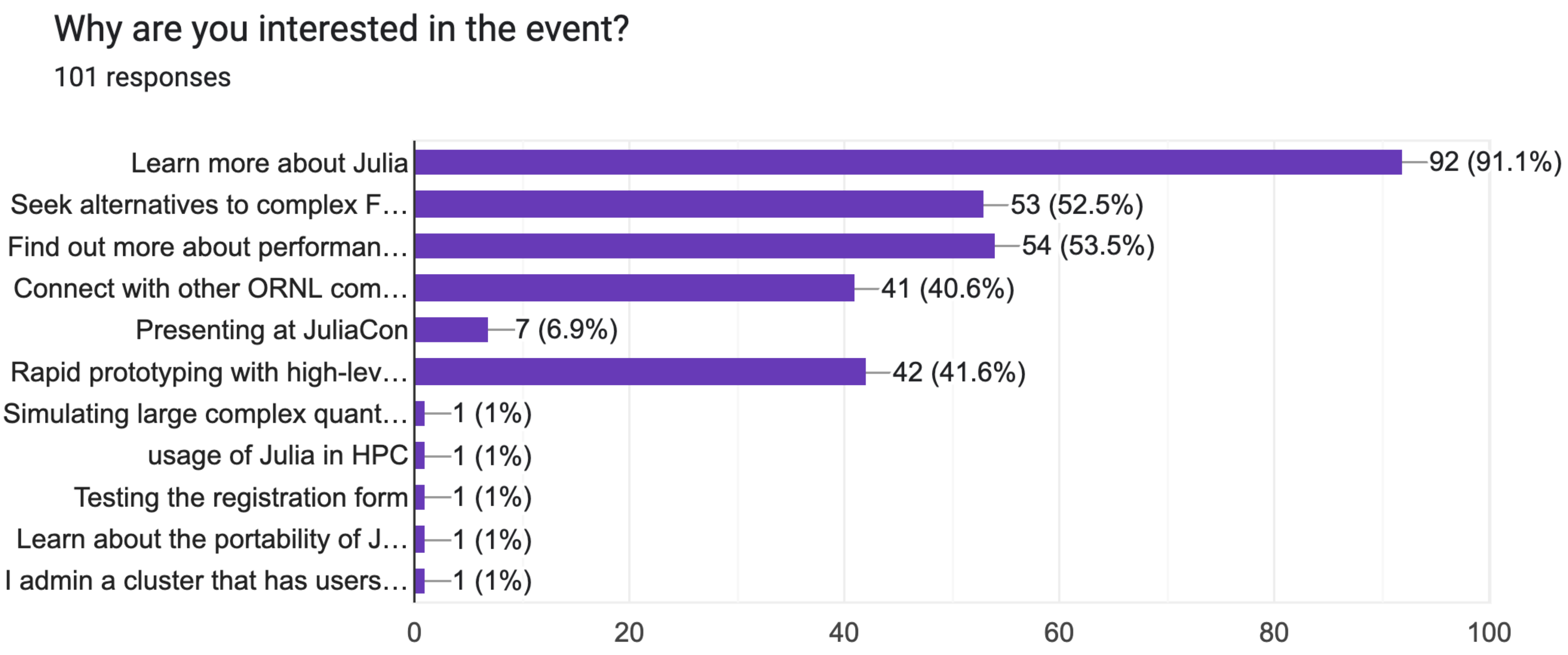
https://williamfgc.github.io/programming/scientific-computing/2020/07/22/first-project-julia-language.html

Contents

- Julia's value proposition for HPC: LLVM + Coordinated Ecosystem
- Community Efforts in HPC
- Running on OLCF System, preliminary results, opportunities
- Resources: where to get started?
- **Final thoughts and acknowledgments**

Final Thoughts

- Results from registered participants at ORNL JuFOS workshop: https://ornl.github.io/events/jufos2022/



Final Thoughts

- Can Julia (as a LLVM DSL for science) help scientists invest more time in science?
- Can Julia help bridging social barriers between AI, data science and HPC?
- Should DOE invest more in a DSL for science, as it did with Fortran, but for our needs in 2022?
- Can performance gaps: JIT, garbage collection, “time-to-first-plot” be leveraged?
- What’s the language and community roadmap, locally and externally?
- What’s the ROI for ORNL scientists adopting Julia?
- Personal reasons:
  - Julia is Fortran for arrays and math ☺ with a rich Standard Library: https://docs.julialang.org/en/v1/
  - LLVM is here to stay for performance. Threads, GPU….fast!
  - No object-oriented, no complex C++ templates, no “dependency hell”
  - I dedicate more time to write tests than figuring out language syntax and environment related bugs
  - Pluto.jl: no messing with conda environments or kernels for Jupyter
  - Software is more “to the point” as expected from a specialized piece of equipment
  - People: community is really active, helpful, and enthusiastic

Acknowledgements

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources of the Oak Ridge Leadership Computing Facility 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.

Thanks to Ross Miller for enabling Julia on Wombat, thanks to Youngsung Kim, Hyun Kang, and Sarat Sreepathi from CSED for sharing their Julia experiences. The Julia for HPC community.

Suzanne Parete-Koon and Michael Sandoval for the invitation.

JuFOS organizers:
- William F Godoy (p)
- Pedro Valero-Lara (p)
- Philip Fackler (p)
- Greg Watson
- Jeff Vetter (p)
- Theresa Ahearn
- Donna Wilkerson

JuFOS Presenters:
- Youngsung Kim
- Ada Sedova
- Gavin Wiggins
- Jean-Luc Fattebert
- Elise Dettling
- Alexia Arthur
- Singanallur Venkatakrishnan
- Christian Trefftz
- John Gounley

JuFOS Sponsors:
The Exascale Computing Project, PROTEAS-TUNE, Proxy App and SRP-HPC sub-projects.

The ASCR Bluestone Project

Thanks to the audience!