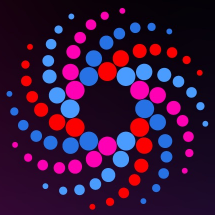
Julia for HPC, SC22 BoF 11/15/2022



SC22
Dallas, TX | hpc accelerates.

William F Godoy[1], Pedro Valero-Lara[1], Valentin Churavy[2], Johannes Blaschke[3], Jeffrey S Vetter[1], Carsten Bauer[4], Mosè Giordano[5]

1 Oak Ridge National Laboratory

2 Massachusetts Institute of Technology

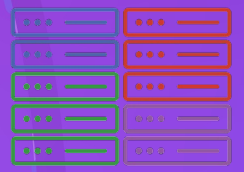
3 NERSC, Lawrence Berkeley National Laboratory

4 Paderborn Center for Parallel Computing, Paderborn University, Germany

5 Centre for Advanced Research Computing, University College London, United Kingdom

Thanks to the U.S. Department of Energy Exascale Computing Project and our multiple sponsors

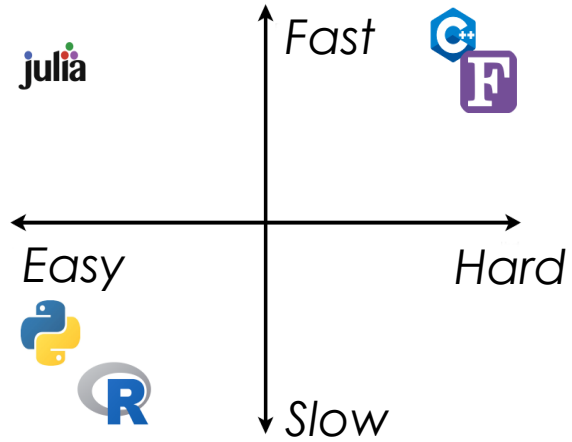
julia



JULIA FOR HPC
Supercomputing 2022 BoF

Goals

- Continue building a diverse "Julia in HPC" multidisciplinary community
- Help us figure it out a vision for: Opportunities, Gaps, Return on Investment, Collaborations
- AI + HPC, post-Moore, heterogeneity, co-design, workflows, reproducibility, better software, accessibility



https://juliadatascience.io/

Background

- Position Paper pre-print: https://arxiv.org/abs/2211.02740

Bridging HPC Communities through the Julia Programming Language

Journal Title
XX(X):1–20

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Valentin Churavy[1], William F Godoy[2], Carsten Bauer[3], Hendrik Ranocha[4], Michael Schlottke-Lakemper[5,6], Ludovic Räss[7,8], Johannes Blaschke[9], Mosè Giordano[10], Erik Schnetter[11,12,13], Samuel Omlin[14], Jeffrey S. Vetter[2], Alan Edelman[1]

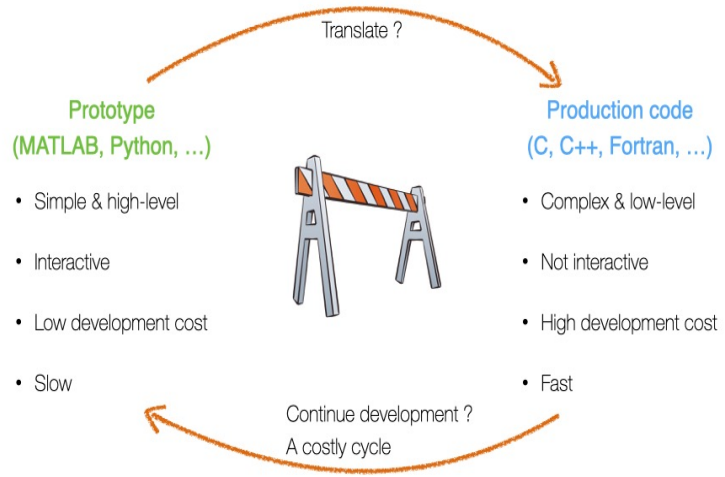
- JuliaCon 2022:
  - Julia for HPC Minisymposium https://www.youtube.com/watch?v=fog1x9rs71Q
  - Julia for HPC BoF https://pretalx.com/juliacon-2022/talk/QVESXM/

"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)

Julia's value proposition for HPC

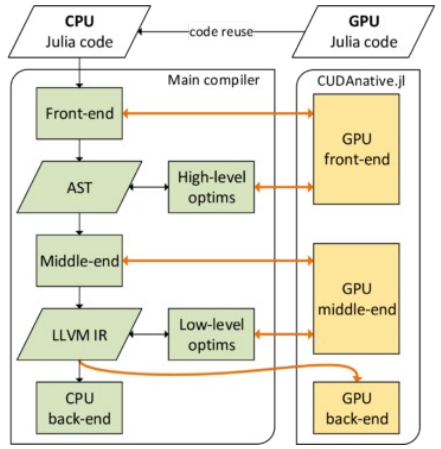
- Designed for "scientific computing" (Fortran) and "data science" (Python) with **performant code via LLVM to access heterogeneous hardware**
- Lightweight **interoperability with existing Fortran and C libraries**
- Julia is a **unifying workflow language** with a **coordinated ecosystem**

"Julia **does not** replace Python/C/C++/Fortran, but the costly process around **Fortran+Python+X**, **C+X**, **Python+X** or **Fortran+X** from ideas to performance portable code for science"
X = { conda, pip, pybind11, cython, Python, C, Fortran, C++, OpenMP, OpenACC, CUDA, HIP, CMake, numpy, scipy, matplotlib, Jupyter, …**}**



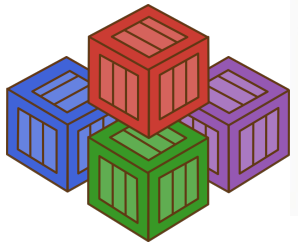
https://pde-on-gpu.vaw.ethz.ch/lecture7

LLVM (widely vendor-adopted)

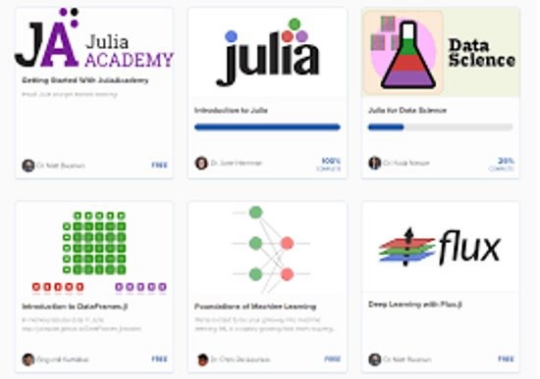


https://developer.nvidia.com/blog/gpu-computing-julia-programming-language/

Pkg.jl



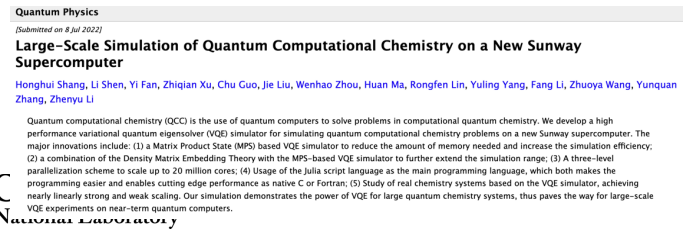
Rich data science ecosystem



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

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

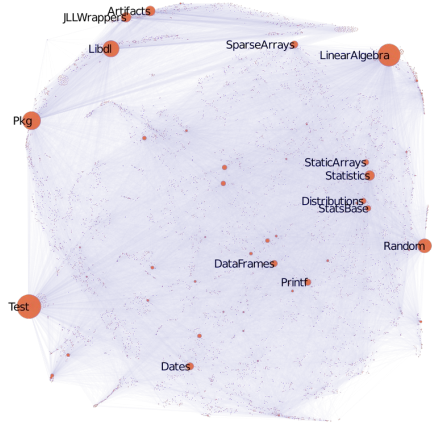
https://enccs.github.io/Julia-for-HPC

https://docs.dftk.org/stable

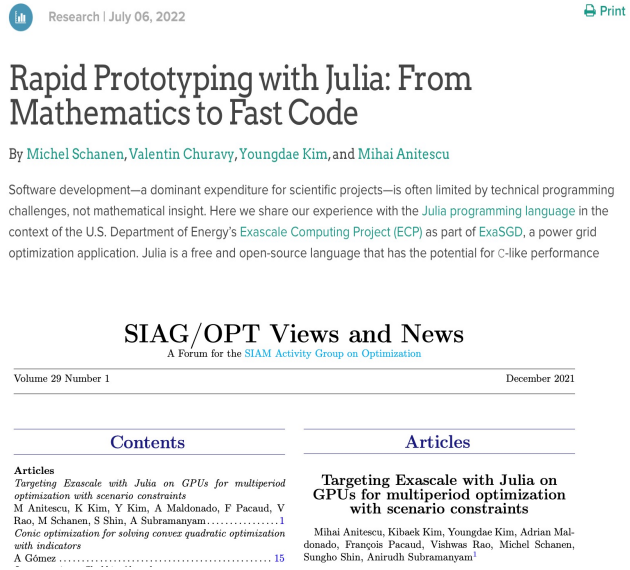
https://juliaastro.github.io/dev

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

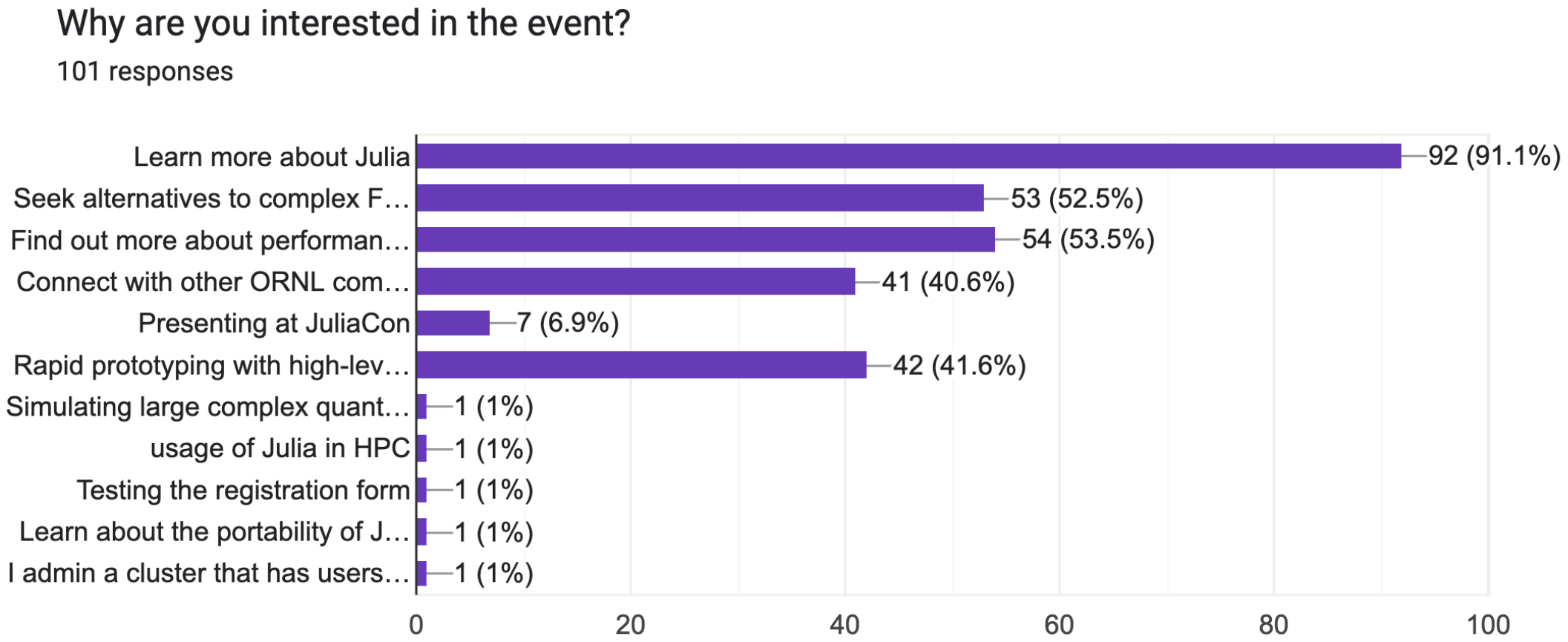
Mihai Anitescu, Kibaek Kim, Youngdae Kim, Adrian Maldonado, François Pacaud, Vishwas Rao, Michel Schanen, Sungho Shin, Anirudh Subramanyam[1]

Today's program

- William: Kickstart/Intro
- Valentin: JuliaLab at MIT
- Johannes: NERSC survey overview
- Carsten: Interactive Julia with Pluto.jl
- Mosè: MPI on Fugaku
- Pedro: OLCF experiences
- Audience interactions, follow up with the community, survey

Final Thoughts

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



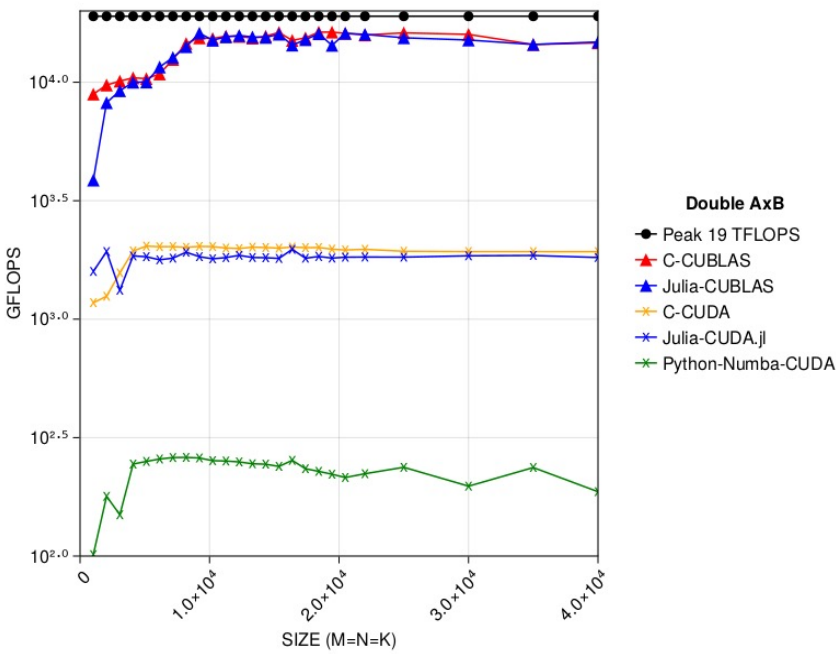
Back up

Performance results on GPU
Results for Matrix Multiplication

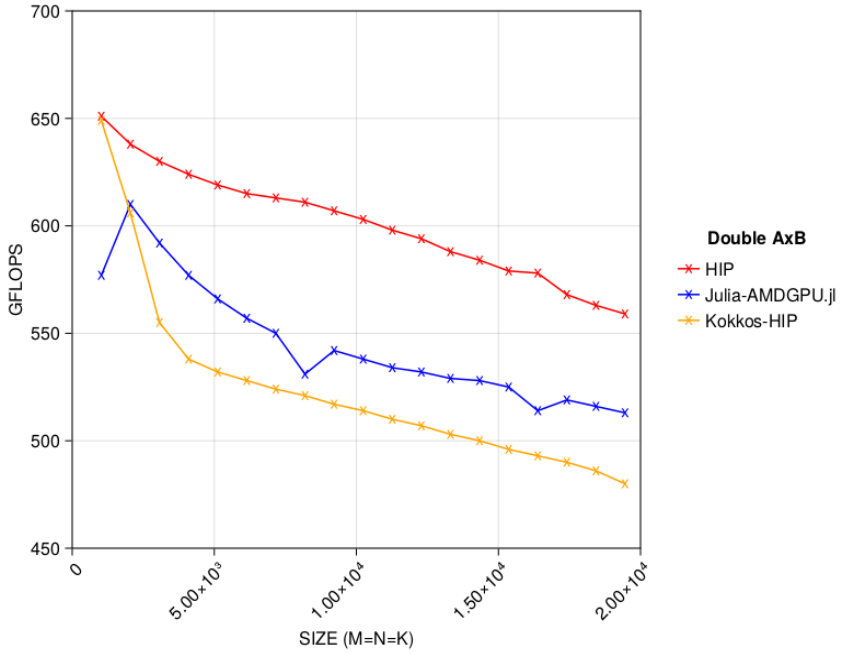
Julia and Python's Numba kernel portability, performance, and productivity on heterogeneous exascale nodes

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

Wombat (ARM)
NVIDIA A100 GPU



Crusher (AMD)
MIX250X



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

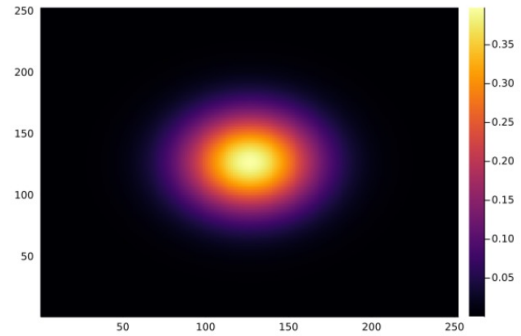
ROCm-MPI

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

OAK RIDGE National Laboratory