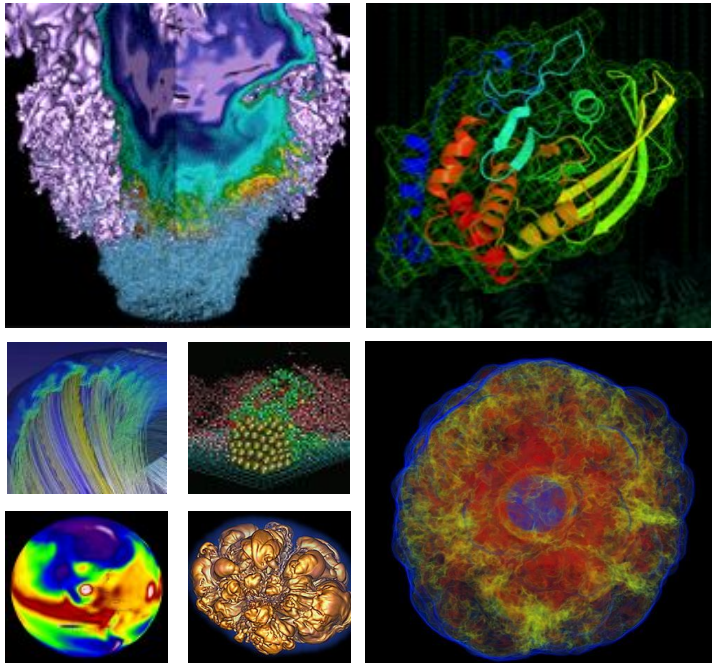
NUG Monthly Meeting



NERSC

25 April, 2023

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Today's plan

- Interactive - please participate!
  - Raise hand or just speak up
  - **NERSC User Slack** (link in chat), **#webinars** channel

- Agenda:
  - Introductions
  - Win-of-the-month
  - Today-I-learned
  - Cori Retirement Update
  - Announcements and Calls for Participation
  - Topic of the day: **Julia at NERSC** with Johannes Blaschke
  - Coming meetings: topic suggestions/requests?

Win of the month

Show off an achievement, or shout out someone else's achievement, e.g.:

- Had a paper accepted
- Solved a bug
- A scientific achievement (maybe candidate for Science Highlight, or **High Impact Scientific Achievement award**)
- An **Innovative Use of High Performance Computing** (also a candidate for an award) (https://www.nersc.gov/science/nersc-hpc-achievement-awards/)

Please let us know of award-worthy work from you or your colleagues - tell us what you did, and what was the key insight?

Today I learned

What surprised you that might benefit other users to hear about?

(and might help NERSC identify documentation improvements!)

Eg:

- Something you got stuck on, hit a dead end, or turned out to be wrong about
  - Give others the benefit of your experience!
  - Opportunity to improve NERSC documentation
- A tip for using NERSC
- Something you learned that might benefit other NERSC users

"If we knew what it was we were doing, it would not be called research, would it?" - Einstein

Cori Retirement Update

Cori is now scheduled to be retired on May 31, 2023 at noon PDT.

Users will still be able to log into the Cori login nodes and access the Cori scratch file system for one week, until Wednesday, June 7, 2023.

To Move Data to Perlmutter:

- Cori Scratch to Perlmutter Scratch: Use Globus (https://docs.nersc.gov/services/globus/)
- Use Community File System for regularly retrieved data
- Use HPSS Tape Archive for infrequently (less frequently) accessed data

Scratch storage is not permanent!

Please back up irreplaceable data to CFS or HPSS!

Cori Retirement Update

For more information, submit a ticket through ServiceNow.

- Next Cori to Perlmutter Office Hours: Tuesday May 2, 10am - 12 noon PST
- See docs.nersc.gov for updated information about using Perlmutter

Seminars, Events, and Trainings*

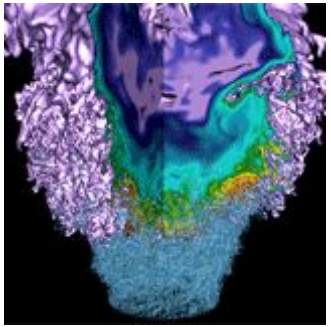
- Codee Training
  - Day 2 tomorrow, April 26th: See NERSC Training Page for information (https://www.nersc.gov/users/training/events/codee-training-series-apr2023/)
- Update PHP before May 17
  - The PHP version on https://portal.nersc.gov will be upgraded from PHP 7.4 to PHP 8.2 on Wednesday, May 17. Users running PHP applications in Portal will need to update to PHP 8 at that time. To easily refactor your PHP code, check out the automatic refactoring tool, Rector. Refactored PHP 8 code can be tested against https://portal-dev.nersc.gov at any time between now and May 17. If you have any questions or concerns about the upgrade timeline, please reach out via ticket.
- ECP Seminar: Lessons Learned Developing Performance-Portable QMCPACK, May 10th
  - presented by Paul Kent (Oak Ridge National Laboratory), Wednesday, May 10, 2023, at 1:00 pm ET; see weekly email for link.

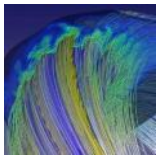
*you can find even more info and events in the weekly emails!

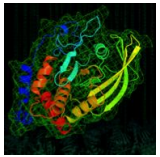
Calls for Participation

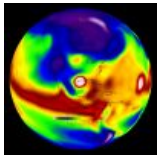
- Student Cluster Competition Info Session
  - Tuesday, April 25, at 9am MDT or on Thursday, April 27, at 6pm MDT. See weekly email for link.
- Fortran User Group LLVM Flang Survey
  - https://forms.gle/vz43AwQyLJ7T47Um7
- NERSC Message of the Day (MOTD) Survey
  - https://forms.gle/fKVHMuCM8wmqzTbDA

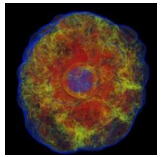
Julia at NERSC

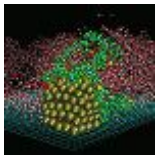












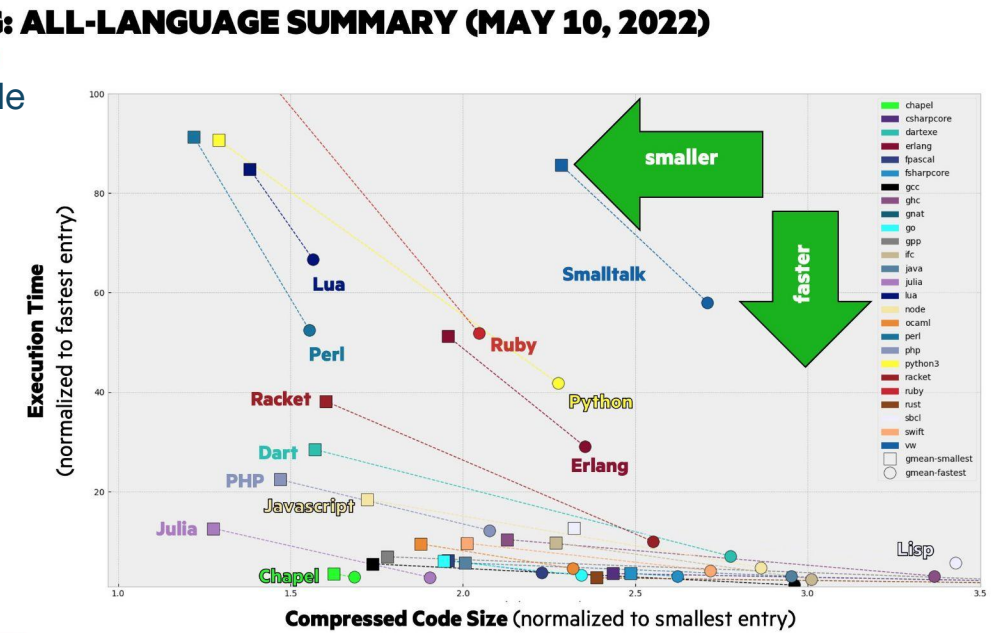
NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

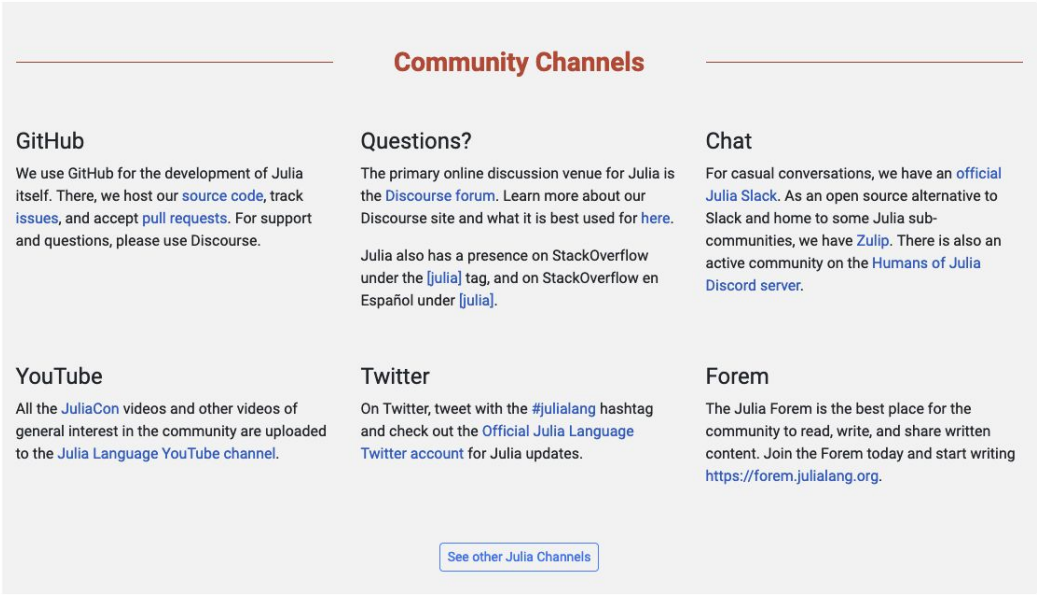
Why Julia?

- Speed with little boilerplate code



Why Julia?

- Speed with little boilerplate code
- Wonderful community: https://julialang.org/community/ (which is HPC-Aware) + https://juliaparallel.org/ (which has monthly calls)



Community Channels

GitHub

We use GitHub for the development of Julia itself. There, we host our source code, track issues, and accept pull requests. For support and questions, please use Discourse.

Questions?

The primary online discussion venue for Julia is the Discourse forum. Learn more about our Discourse site and what it is best used for here.

Julia also has a presence on StackOverflow under the [julia] tag, and on StackOverflow en Español under [julia].

Chat

For casual conversations, we have an official Julia Slack. As an open source alternative to Slack and home to some Julia sub-communities, we have Zulip. There is also an active community on the Humans of Julia Discord server.

YouTube

All the JuliaCon videos and other videos of general interest in the community are uploaded to the Julia Language YouTube channel.

Twitter

On Twitter, tweet with the #julialang hashtag and check out the Official Julia Language Twitter account for Julia updates.

Forem

The Julia Forem is the best place for the community to read, write, and share written content. Join the Forem today and start writing https://forem.julialang.org.

See other Julia Channels

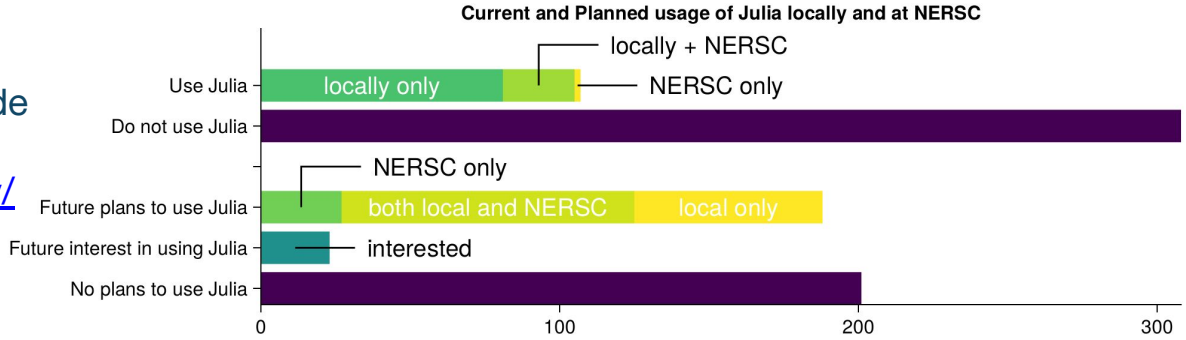
Why Julia?

- Speed with little boilerplate code
- Wonderful community: https://julialang.org/community/ (which is HPC-Aware) + https://juliaparallel.org/ (which has monthly calls)
- Low barrier between frontend and backend

Function signature	Pybind11		ccall		speedup
`int fn0()`	132	±14.9	2.34	±1.24	56×
`int fn1(int)`	217	±20.9	2.35	±1.33	92×
`double fn2(int, double)`	232	±11.7	2.32	±0.189	100×
`char* fn3(int, double, char*)`	267	±28.9	6.27	±0.396	42×

Why Julia?

- Speed with little boilerplate code
- Wonderful community: https://julialang.org/community/ (which is HPC-Aware) + https://juliaparallel.org/ (which has monthly calls)
- Low barrier between frontend and backend
- Lots of NERSC (and HPC) users are already using (or planning to use) Julia as part of their work



Bridging HPC Communities through the Julia Programming Language

Journal Title
XX(X):1–20

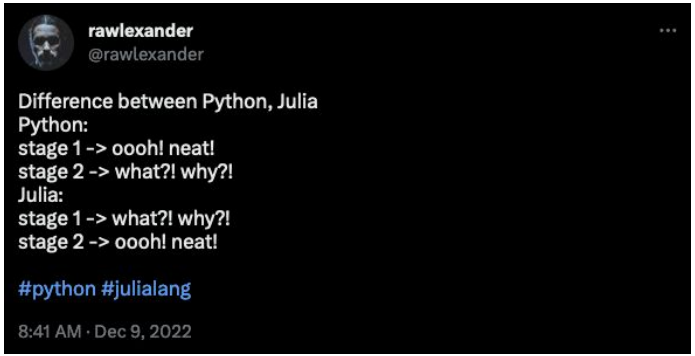
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

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]

https://arxiv.org/abs/2211.02740

Why Julia?

- Speed with little boilerplate code
- Wonderful community: https://julialang.org/community/ (which is HPC-Aware) + https://juliaparallel.org/ (which has monthly calls)
- Low barrier between frontend and backend
- Lots of NERSC (and HPC) users are already using (or planning to use) Julia as part of their work
- Great language features



Julia News

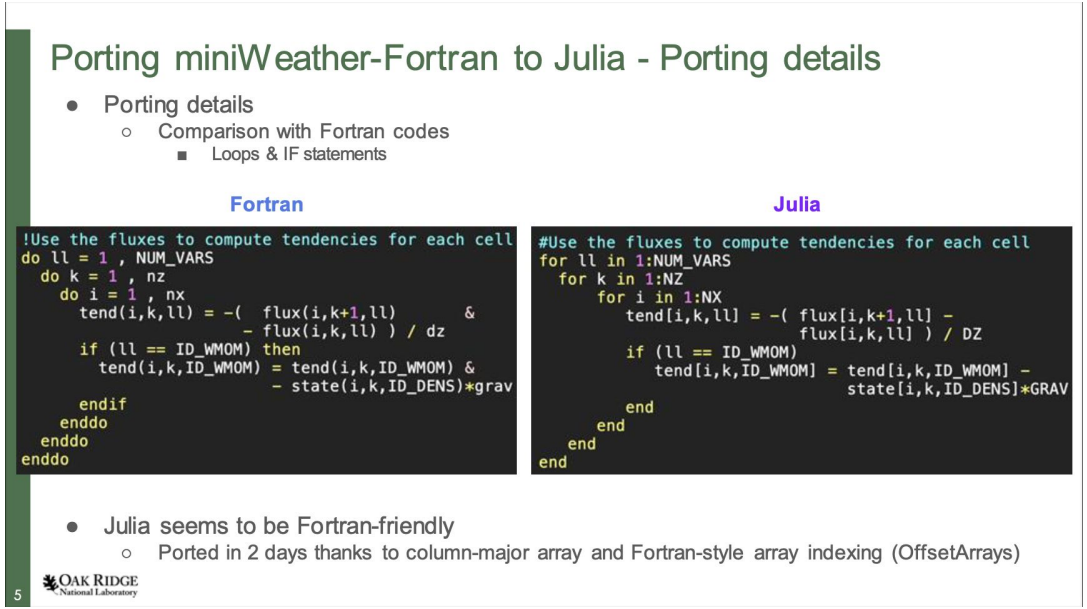
- The current Julia modules on PM are a bit broken
  - (Even though the language and API is stable), the Julia ecosystem is too fast moving for the regular HPC module release cycle
  - Moving away from NERSC-provided environments
  - Making it easier for users to set up and maintain their own environments
- MPI.jl + GTL Hacks will soon no longer be needed:
  - https://github.com/JuliaParallel/MPI.jl/pull/716
  - Install using:

    ```
    MPIPreferences.use_system_binary(; vendor="cray", mpiexec="srun")
    ```
- Juliaup is now HPC-friendly
  - https://github.com/JuliaLang/juliaup/pull/582
- Distributed HSN affinity
  - WIP (using HWLOC.jl to find the “right” HSN NIC)
  - In the meantime, use this workaround: `JULIA_DEPOT_PATH[2]/packages/common/bin/julia_hsn_launcher.sh`

    ```
    export NERSC_NODE_HSN_IP=$(ip a show dev hsn0 | grep global | cut -d' ' -f6 | cut -d'/' -f1)
    julia --bind-to=$NERSC_NODE_HSN_IP "$@"
    ```

Training Materials

- Background and HPC Examples: https://github.com/JuliaParallel/julia-ecp-community-days-2023
- Upcoming: https://www.olcf.ornl.gov/calendar/julia-for-high-performance-computing-tutorial/



Porting miniWeather-Fortran to Julia - Porting details

- Porting details
  - Comparison with Fortran codes
    - Loops & IF statements

Fortran

```
!Use the fluxes to compute tendencies for each cell
do ll = 1 , NUM_VARS
  do k = 1 , nz
    do i = 1 , nx
      tend(i,k,ll) = -(  flux(i,k+1,ll)       &
                       - flux(i,k,ll) ) / dz
      if (ll == ID_WMOM) then
        tend(i,k,ID_WMOM) = tend(i,k,ID_WMOM) &
                          - state(i,k,ID_DENS)*grav
      endif
    enddo
  enddo
enddo
```

Julia

```
#Use the fluxes to compute tendencies for each cell
for ll in 1:NUM_VARS
  for k in 1:NZ
      for i in 1:NX
          tend[i,k,ll] = -( flux[i,k+1,ll] -
                            flux[i,k,ll] ) / DZ
          if (ll == ID_WMOM)
             tend[i,k,ID_WMOM] = tend[i,k,ID_WMOM] -
                                 state[i,k,ID_DENS]*GRAV
          end
      end
  end
end
```

- Julia seems to be Fortran-friendly
  - Ported in 2 days thanks to column-major array and Fortran-style array indexing (OffsetArrays)

5 OAK RIDGE National Laboratory

How to Manage Your Julia Project

- Package management (Jupyter Demo)

- HPC Considerations: Containers + Compile cache file system
  - If you can, use PackageCompiler.jl + podman-hpc to pre-compile as much of your project into the image. This might not always be feasible => mount the compile cache (on scratch) into the image
  - `podman-hpc run -v $SCRATCH/julia-compile-cache:/mnt/julia-compile-cache …`
  - (In the image):
    `export JULIA_DEPOT_PATH=/mnt/julia-compile-cache:$JULIA_DEPOT_PATH`
  - Compile cache is kept at: `DEPOT_PATH[1]/compiled`

- Further reading: **PackageCompiler.jl** https://julialang.github.io/PackageCompiler.jl/stable/
  - Precompiles packages and creates sysimages from Julia packages

Demo: Julia from Jupyter at NERSC

- Distributed Example on Jupyter

Noteworthy Julia Packages (for HPC)

- **JuliaIO**: https://github.com/JuliaIO
- **JuliaData**: https://github.com/JuliaData
  Collects many Julia packages around I/O and Data
  - **Tables.jl** / **DTables.jl** / **DistributedArrays.jl**: arrays and tables build on distributed / **CSV.jl**: Tabular data support
  - **DTables.jl** / **DistributedArrays.jl**: arrays and tables build on distributed

- **JuliaParallel**: https://github.com/JuliaParallel
  Collects many Julia packages around distributed and parallel computing
  - **MPI.jl**: no explanation needed (it is CUDA/ROCM-aware)
  - **ClusterManagers.jl**: manager HPC resources on the fly (also note **SlurmClusterManager.jl** and **MPIClusterManagers.jl** for HPC clusters)
  - **ImplicitGlobalGrid.jl** / **MPIArrays.jl**: implement a global address space (using the Array interface) built on MPI.jl
  - **Distributed.jl** / **Dagger.jl**: task-based parallelism (like Dask and Ray)

- **JuliaGPU**: https://github.com/JuliaGPU
  Collects many Julia packages used for GPU computing

- ML support: **Flux.jl** (like pytorch, but different)

Coming up

Upcoming topics:

- JupyterHub at NERSC
- NERSC Tips and Tricks
- Security at NERSC

We'd love to hear more lightning talks **from NERSC users** about the research you use NERSC for!

Nominate a topic at: https://forms.gle/WjYx7zV7SAz2CaYz7

Science Highlights Submission: https://docs.google.com/forms/d/e/1FAIpQLScP4bRCtcde43nqUx4Z_sz780G9HsXtpecQ_qIPKvGafDVVKQ/viewform

Lightning Talks

Highlights

Thank You