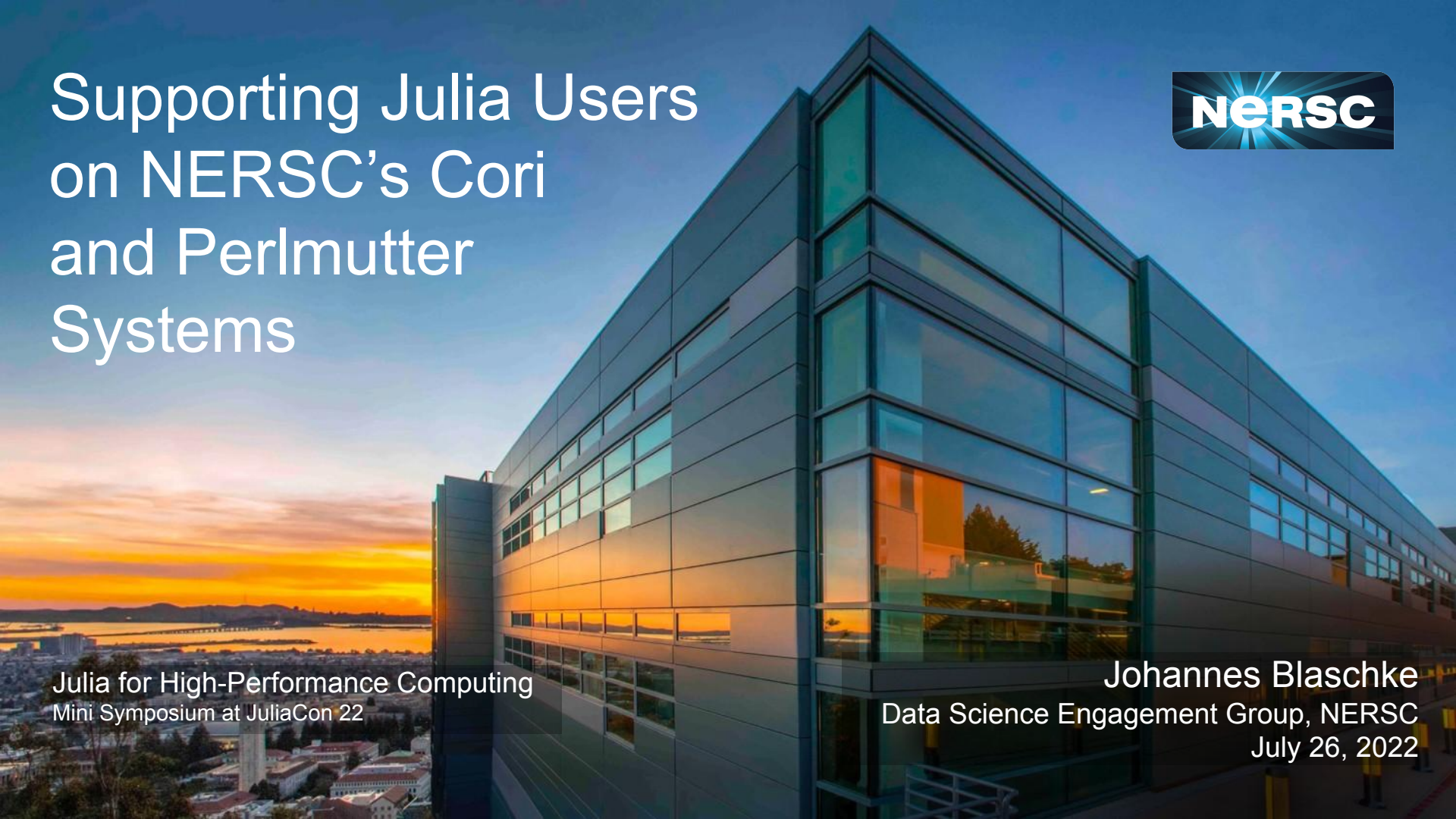
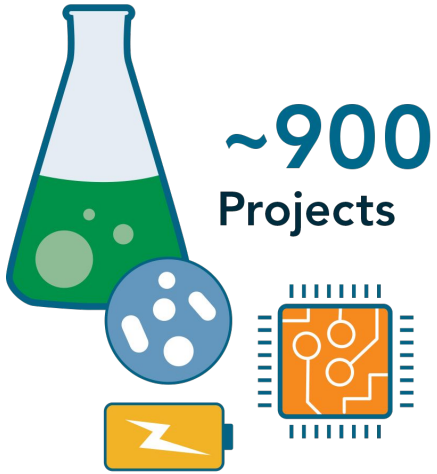
Supporting Julia Users on NERSC's Cori and Perlmutter Systems



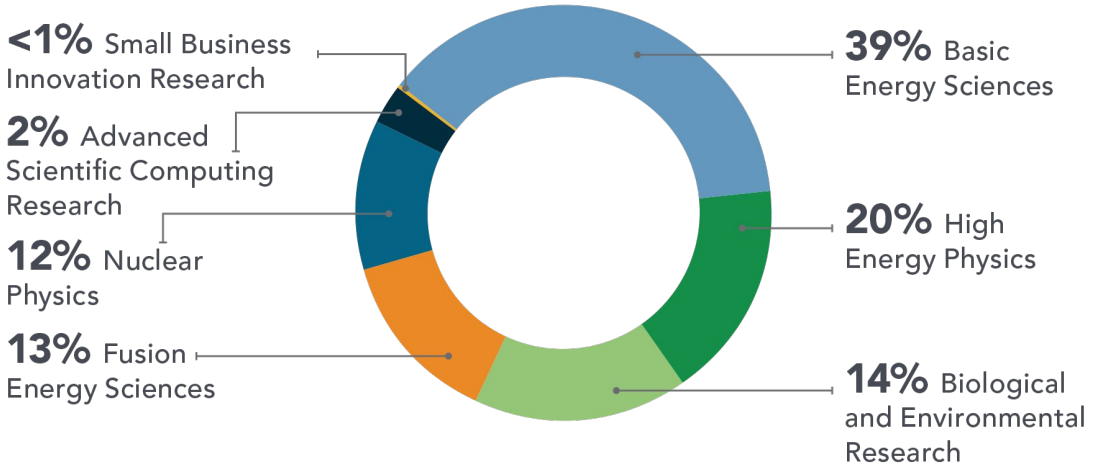
Julia for High-Performance Computing
Mini Symposium at JuliaCon 22

Johannes Blaschke
Data Science Engagement Group, NERSC
July 26, 2022

2020 NERSC by the Numbers

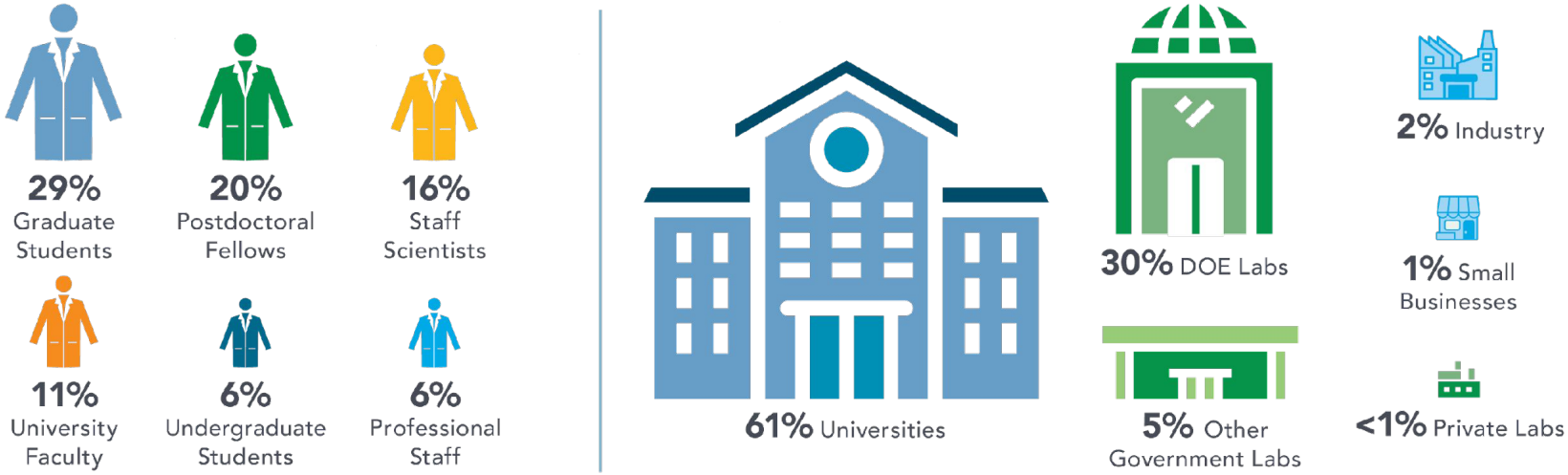


2020 DOE Office of Science Program Usage Breakdown

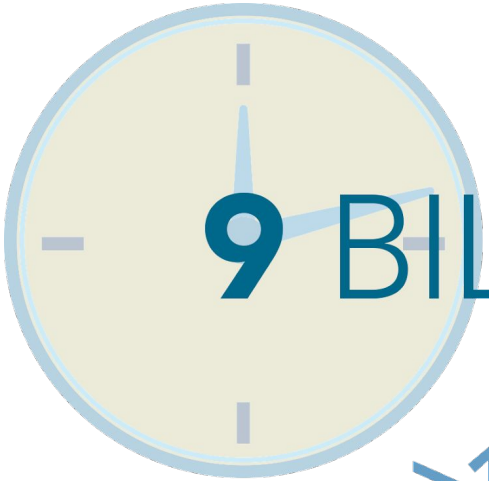


2020 NERSC by the Numbers

~8,000 ANNUAL USERS FROM ~1,750 Institutions + National Labs

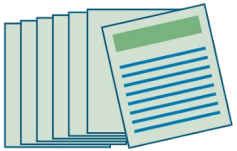


2020 NERSC by the Numbers



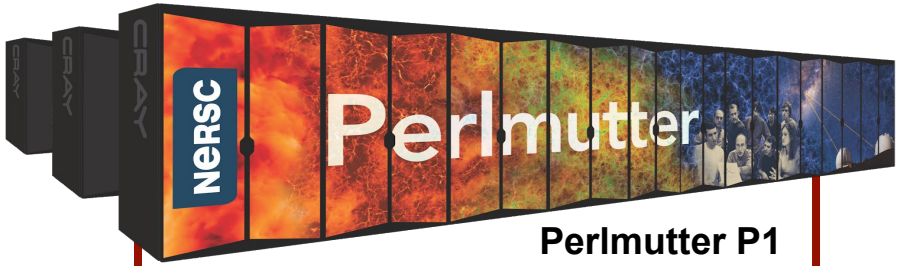
9 BILLION CORE HOURS USED IN 2020

>1,800
Refereed Publications
Cited NERSC



Data Stored
200
Petabytes

NERSC Systems Spring 2022

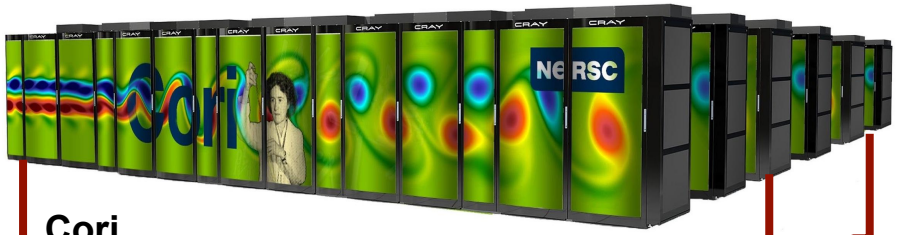


Perlmutter P1

1,536 NVIDIA A100 accelerated nodes
4 A100 GPUs & 1 AMD 'Milan' CPU per node
384 TB (CPU) + 240 TB (GPU) memory
HPE Cray Slingshot high speed interconnect
World's 5th most powerful supercomputer
140 PF Peak
Pre-production system

5 TB/s

35 PB Scratch



Cori

9,600 Intel Xeon Phi "KNL" manycore nodes
2,000 Intel Xeon "Haswell" nodes
700,000 processor cores, 1.2 PB memory
Cray XC40 / Aries Dragonfly interconnect
30 PF Peak

1.5 TB/s

2 PB Burst Buffer

700 GB/s

28 PB Scratch

50 GB/s

HPSS Tape Archive ~200 PB

DTNs, Spin, **Jupyter**, Gateways, **Workflow Nodes**

ESnet ENERGY SCIENCES NETWORK

2 x 100 Gb/s SDN

Ethernet & IB Fabric
Science Friendly Security
Production Monitoring
Power Efficiency
LAN

100 GB/s

120 PB Common File System

5 GB/s

275 TB /home

NERSC TL;DR

- Our Users come to NERSC with varying amounts of HPC experience and with a very broad range of science problems
- Our hardware is heterogeneous
- Future high-profile engagements will increasingly involve cross-facility engagements (e.g. live data processing from experiments)

- **Need #1**: High-Productivity High-Performance Glue Code
- **Need #2:** Portable (Whole) Workflow Performance

The Need for High-Performance Glue Code

- **Objective**: Establish High-Productivity High-Performance Programming Languages
- Common Design Pattern: High-Productivity Language (eg. Python) as Glue Code
  - At NERSC: Julia, Python + C/C++/CUDA
  - Pro: Use appropriate language for algorithms requiring high performance
  - Con: N+1-language problem (code maintainability)
  - Con: Context switching between interpreted and compiled languages

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×

Julia Usage Trends at NERSC

- Growing interest in Julia at NERSC:

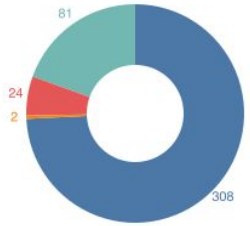
HPC wire

Julia Joins Petaflop Club

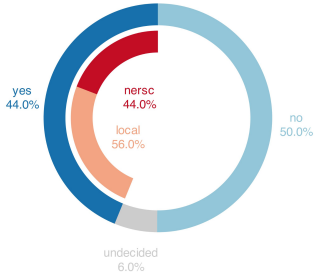
September 12, 2017

BERKELEY, Calif., Sept. 12, 2017 — ,

Do you use Julia locally or at NERSC?	Responses	%
"I do not use Julia (locally or at NERSC)"	308	74
"I use Julia locally but not at NERSC"	81	20
"I use Julia locally and at NERSC"	24	6.8
"I use Julia at NERSC but not locally"	2	0.5



Do you plan to use Julia in future?



Julia Support at NERSC

- **Objective**: Enable users to “roll their own” Julia install / environment
- Support different “levels” of Julia users:
  a. Provide documentation and use cases
  b. Provide system-wide settings
     (user can load this module,
     but doesn’ need to use our depot)
  c. Provide compatibility interfaces, eg. `MPItrampoline`
  d. Modules include pre-compiled packages
     in the `JULIA_DEPOT_PATH` and
     `JULIA_LOAD_PATH`

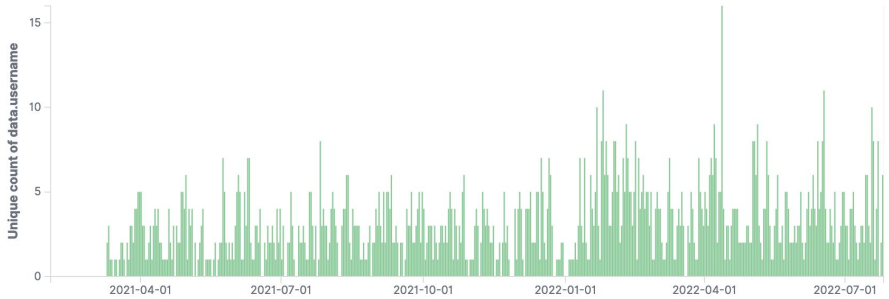
```
35 ## Software-specific settings exported to user environment
36 setenv JULIA_CUDA_USE_BINARYBUILDER false
37 setenv JULIA_MPI_BINARY             system
38 setenv JULIA_MPI_PATH               $env(CRAY_MPICH_DIR)
39 setenv JULIA_MPIEXEC                srun
```

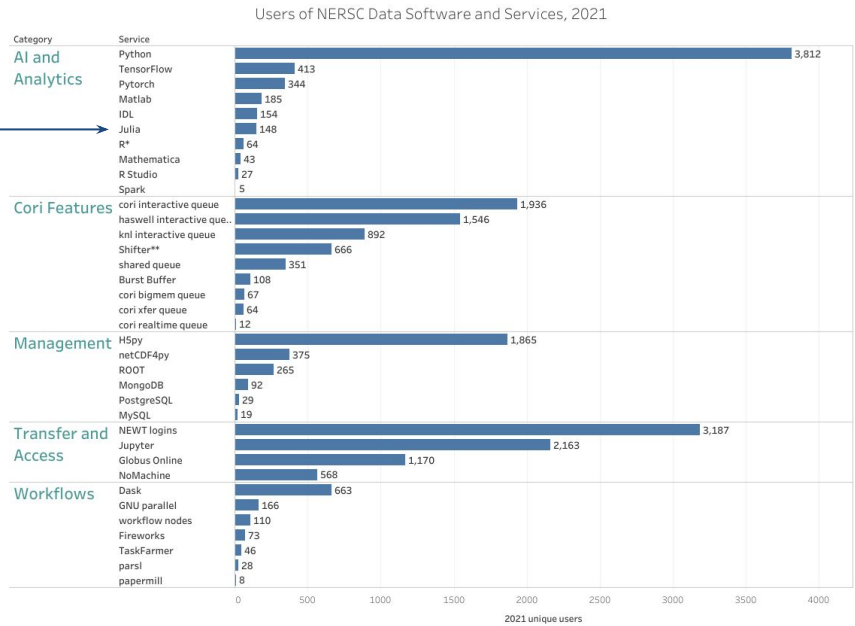
```
30 ## Software-specific settings exported to user environment
31 module      load            julia/settings-$mpich_compiler
32 prepend-path PATH            $root/bin
33 prepend-path PATH            $admin_depot/bin
34 prepend-path JULIA_DEPOT_PATH $env(HOME)/.julia/nersc/$platform:$pkg_depot
35 setenv       JULIA_ADMIN_PATH $admin_depot
36 prepend-path JULIA_LOAD_PATH  "@:@v#.#:$admin_depot/environments/globalenv:@stdlib"
```

Julia Usage Trends at NERSC

- Growing use of Julia modules at NERSC

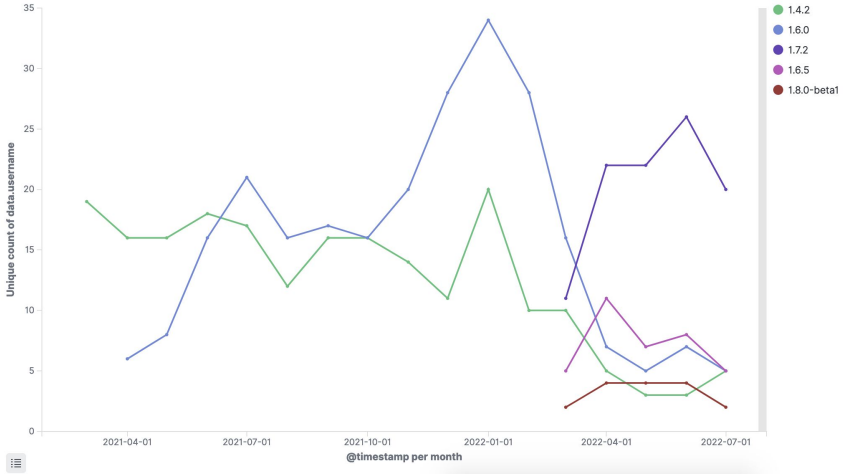
229 module users
(04/21-07/22)





Julia Usage Trends at NERSC

- Julia Users like new versions
- Difficult for center software release cycle to keep up with latest Julia version
  - Use CI/CD to keep up to date
  - Enable users to be productive with their own Julia versions



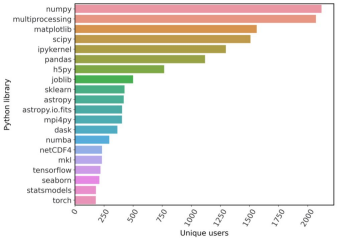
Ongoing and Future Work

- **Detailed Usage Monitoring:** Use `startup.jl` to register `atexit` hook which monitors loaded packages
- **Production-Level Support:** Optimize Julia performance on NERSC systems and integrate support into center operations
- **Advanced Workflow Control:** Explore how workflow managers interact with center resource scheduler (eg. Slurm) *in situ* using API (eg. PMI2)
- **Documentation, Use Cases, and Training**

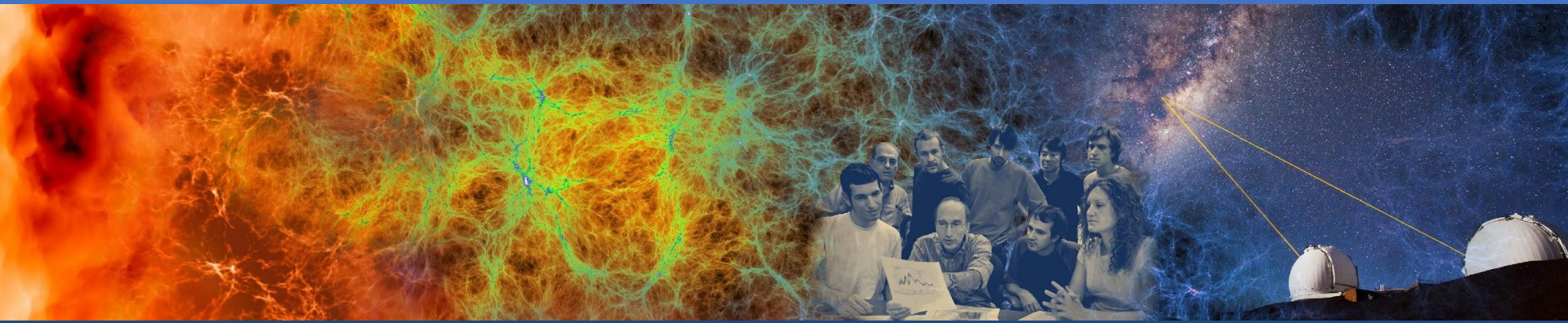
PROC. OF THE 20th PYTHON IN SCIENCE CONF. (SCIPY 2021)

Monitoring Scientific Python Usage on a Supercomputer

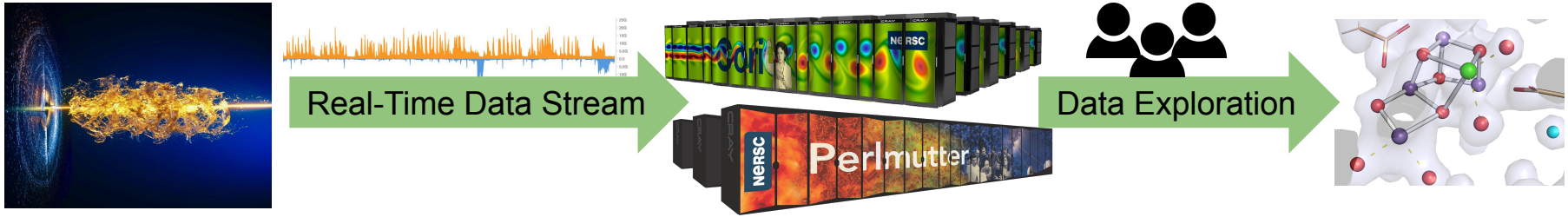
Rollin Thomas[‡*], Laurie Stephey[‡*], Annette Greiner[‡], Brandon Cook[‡]



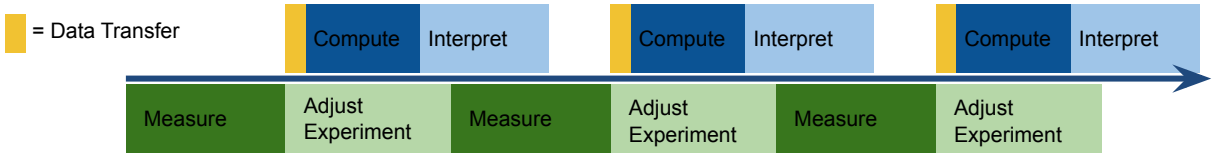
Extra Slides



Rapid Prototyping Case Study: Real-Time Data Analysis



- **Challenge:** Reconstruct (unknown) molecular structures from (never before seen) X-Ray scattering data by analyzing pixel-level data
  - While keeping up with experiment => Analyze O(TB) in 30 mins



- **Solution:** Plug a Supercomputer into your experiment (and drive it from Jupyter in real-time?)

NESAP Applications Cover the Broad Workload

Electronic Structure	
Quantum ESPRESSO	BES
NWChemEX	BES
VASP	BES
MFDn	NP
WEST	BES
BerkeleyGW	BES

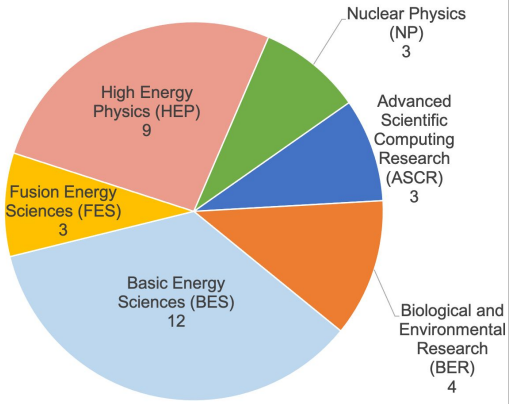
Molecular Dynamics	
EXAALT	FES, NP, BES
NAMD	BES, BER

Data	
DESI	HEP
TomoPy	BES
ATLAS	HEP
ExaFel	BES, ECP
CMS	HEP
ExaBiome	BER, ECP
TOAST	HEP
FICUS	BER
LZ	HEP

Learning	
ExaRL	BES
HEP Accel ML	HEP
Catalyst ML	BES
Extreme Spatio-Temporal ML	ASCR
FlowGAN	ASCR

Particles & Grids	
ASGarD	FES, ASCR
WarpX	HEP, ECP
ImSim	HEP
ChomboCrunch	BES, ECP
E3SM	BER, ECP
WDMAPP	FES, ECP

LQCD	
LQCD Consortium	HEP, NP



+29 Tier 2 NESAP teams

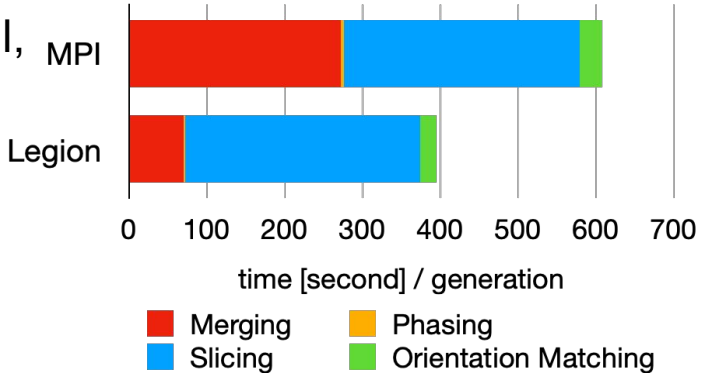
58 Total NESAP Teams

Future Julia Support at NERSC

- Add Perlmutter support
  ⇒ Currently experimental
  - Early user testing found issues that require a fix to Julia (currently have workaround)
- Julia dev is fast-moving
  ⇒ Relying on NERSC-provided modules is not enough
- Speed up module release cycle (some automation is involved)
- Add Pkg usage monitoring

The Need for Workflow Performance

- High-productivity languages allow for rich programming models, making them as ideal to express workflow “contracts” (eg. DAG)
- At NERSC: Dask, Legion, Parsl, Nextflow, Papermill, Fireworks, Taskfarmer, Snakemake, GNU Parallel + Jupyter Widgets!
- **Common Design Pattern**: User starts server (on workflow node) and requests resources, workers (on compute nodes) connect to resources vis HSN



- **Missing**: Very few workflow managers interact with center resource scheduler (eg. Slurm) *in situ* using API (eg. PMI2) – limiting flexibility

Workflow Support Story

- This code was about 100x slower on Perlmutter

```
using Distributed, ClusterManagers

using Sockets

em = ElasticManager(addr=:auto, topology=:master_worker, port=9009)

println(em)

# launch workers
@async run(`srun -n $NWORKERS sh -c $(ClusterManagers.get_connect_cmd(em))`)

# wait for them to connect
while nworkers()<NWORKERS
    sleep(1)
end

# test transfer
y = rand(Float32, 4*1024^2)
for i=1:20
    @time @everywhere x = $y
end
```

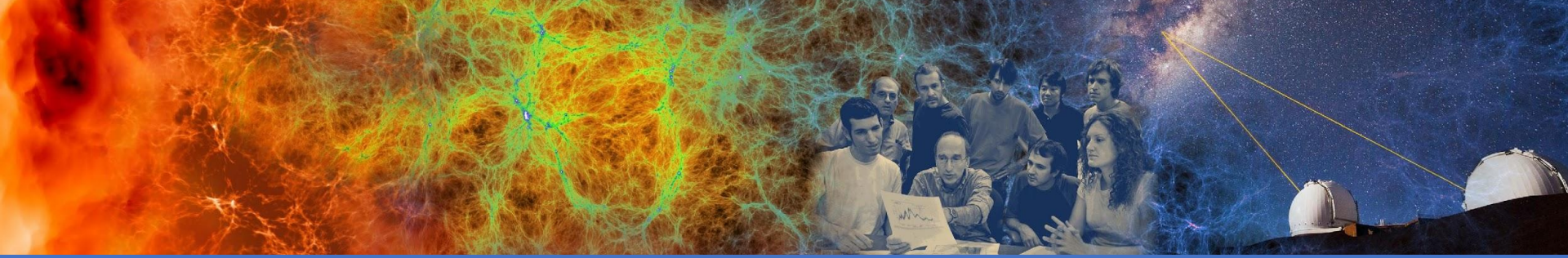
- Investigated the cause:
  On Perlmutter, `Sockets` uses the node management network (rather than the high-speed network) unless we hack `Sockets`:

```
try
    using Sockets
    perlmutter_hsn_addr = IPv4(only(filter(!isnothing, match.(r"inet (.*)/.*hsn0:border", readlines(`ip a show`)))).captures[1])
    Sockets.getipaddr() = perlmutter_hsn_addr
catch
end
```

NUG Involvement in Julia Support

1. NUG: Help other users to more easily “roll their own” Julia install / environment by:
   - Help document best practises community use cases
   - Connecting and exchanging knowledge with other Julia users
   - Provide feedback to NERSC staff (to help with #2)

2. NERSC: Enable users to more easily “roll their own” Julia install / environment by:
   - Providing a venue for #1 / Help organize training
   - Decoupling NERSC-configuration from Julia module
   - Technical support to interface Julia runtime with a Supercomputer (eg. BLAStrampoline.jl, MPItrampoline.jl)



The End