Hewlett Packard
Enterprise

Accelerating scientific research through high performance computing democratization

Andrew Shao, PhD
Senior Software Engineer, Master Technologist, HPE
Scott Bachman, PhD
Research Scientist, National Center for Atmospheric Research
Visiting Research Scholar, HPE

15 February 2023

ABOUT Andrew

Background:

- Oceanography, computational physics, and applied mathematics
  - Brief stints in economics and philosophy
- Specialties: Numerical methods, climate modelling, combining HPC and AI

Relevant Interests:

- Community-based scientific software development
- Combining numerical and data-based approaches
- Building complex AI/ML and simulation workflows

Roles:

- Senior HPC & AI Research Scientist, HPE Canada

Hewlett Packard Enterprise

ABOUT Scott

Background:

- Climate Scientist, National Center for Atmospheric Research (Boulder, CO)
- Specialties: Physical Oceanography, model development, turbulence

Relevant Interests:

- High performance computing for Earth system prediction
- Exascale data analysis
- Democratization of science and scientific tools

Current role:

- Visiting Scholar, Hewlett Packard Enterprise (Chapel Development Team)

NCAR

Hewlett Packard Enterprise

Outline

- Scientific computing – the individual's perspective
- Scientific computing – the institutional perspective
- Open-sourcing AI/ML with SmartSim
- Open-sourcing distributed computing with Chapel/Arkouda

Scott’s haphazard education in computing

- Exposed to **C++** (undergrad, ca. 2003)
- Learned **MATLAB** in first course on scientific programming (grad school, ca. 2007)
- First interaction with HPC (2008)
- First interaction with MPI / OpenMP (2008)
- Started learning **Python** (2009)
  - ”Python is the way of the future”
  - MATLAB not open source
- First encounter with **C** and **Fortran** (2015)
- Inundation with Fortran (2017, start of current job)
- First encounter with Dask (2019)
- **Chapel** (2022)

Career turned toward computational science (fluid dynamics)

Career turned toward climate science

At any given career stage, I only learned the language(s) I **needed** for my job.

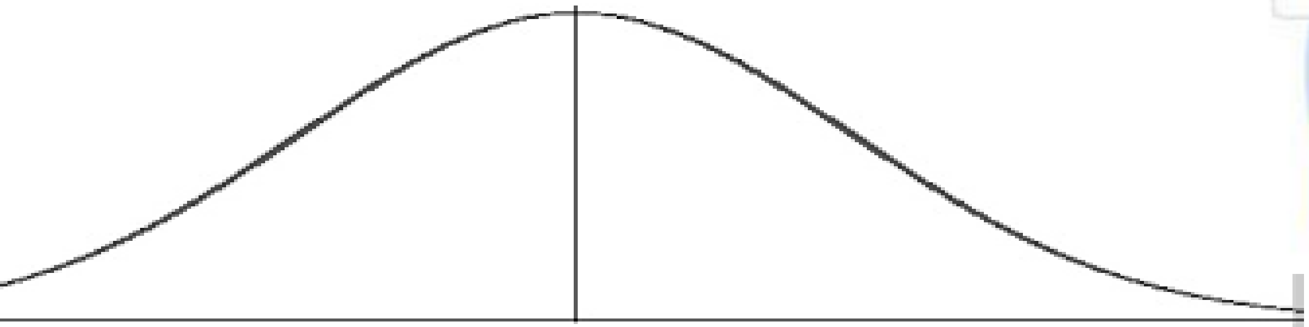
Scientists and Computers: Fun Facts

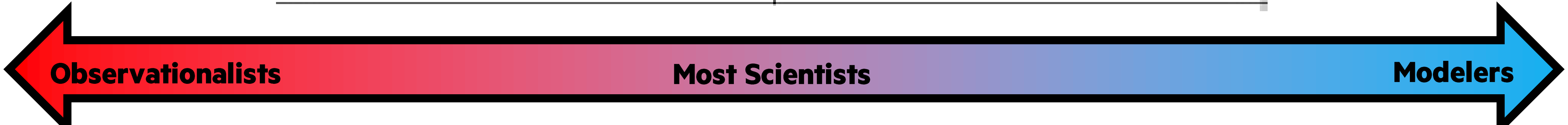
1. Scientists' computing skills and comfort level vary dramatically.
2. Scientists are curious about new computing technologies.
3. Scientists are suspicious about new computing technologies.
4. Scientists strongly prefer the "show" in "show-and-tell".
5. Most scientists are easily waylaid by DIY software setup.
6. Scientists develop a strong loyalty to solutions that "just work".
7. Scientists develop a strong loyalty to languages and programs.
8. Scientists develop inertia against learning new solutions, languages, and programs.
9. Scientists don't have the bandwidth to seek incremental performance improvements on their own.
10. Scientists' careers are shaped by the computing solutions that are available to them.
11. Scientists LOVE software engineers.

Scientists and Computers: Fun Facts

1. Scientists' computing skills and comfort level vary dramatically.

   - Communication has to be adjusted for the audience.
   - Will this scientist **appreciate** what your solution is offering to them, or the solution itself?





Observationalists

Most Scientists

Modelers

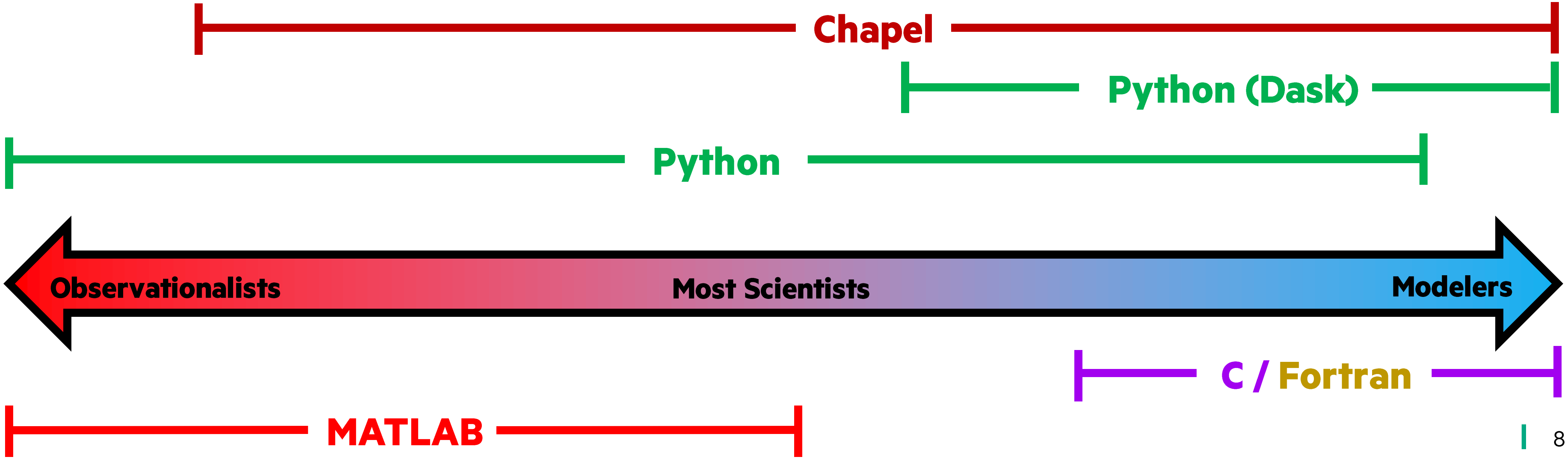
Uncomfortable!

Comfortable!

Scientists and Computers: Fun Facts

1. Scientists' computing skills and comfort level vary dramatically.

   - Communication has to be adjusted for the audience.
   - Will this scientist **appreciate** what your solution is offering to them, or the solution itself?



Scientists and Computers: Fun Facts

2. Scientists are curious about new computing technologies.

    - Scientists enjoy learning!
    - Scientists think **many** things are “cool” or “interesting”.
    - Scientists are **ALWAYS** looking for better / faster / easier ways to do their work.

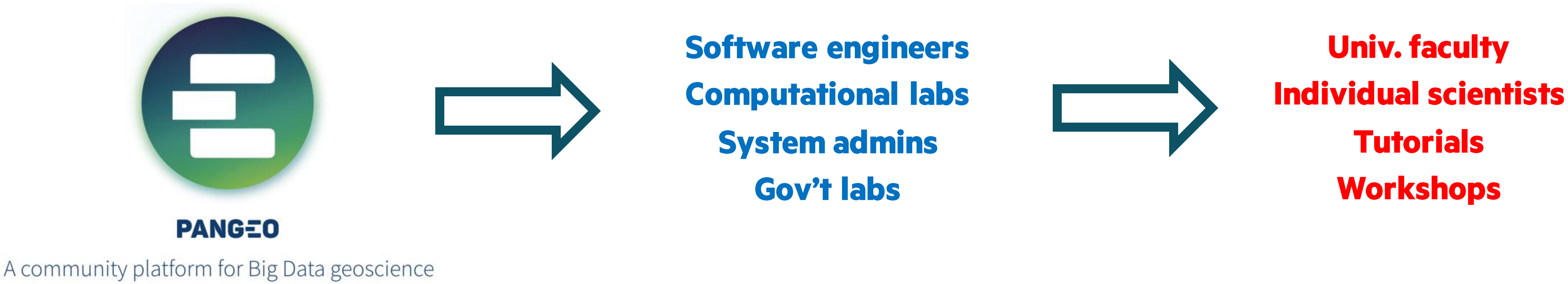
Cheering at NASA Mission Control after Mars Perseverance landing.

Scientists and Computers: Fun Facts

3. Scientists are suspicious about new computing technologies.

   - Opportunity cost?
   - Scientists are wary of investing time + effort.
   - Is it hype? Or is it legit?
   - Solutions often “trickle down” from specialists to non-specialists.

Example:



PANGEO

A community platform for Big Data geoscience

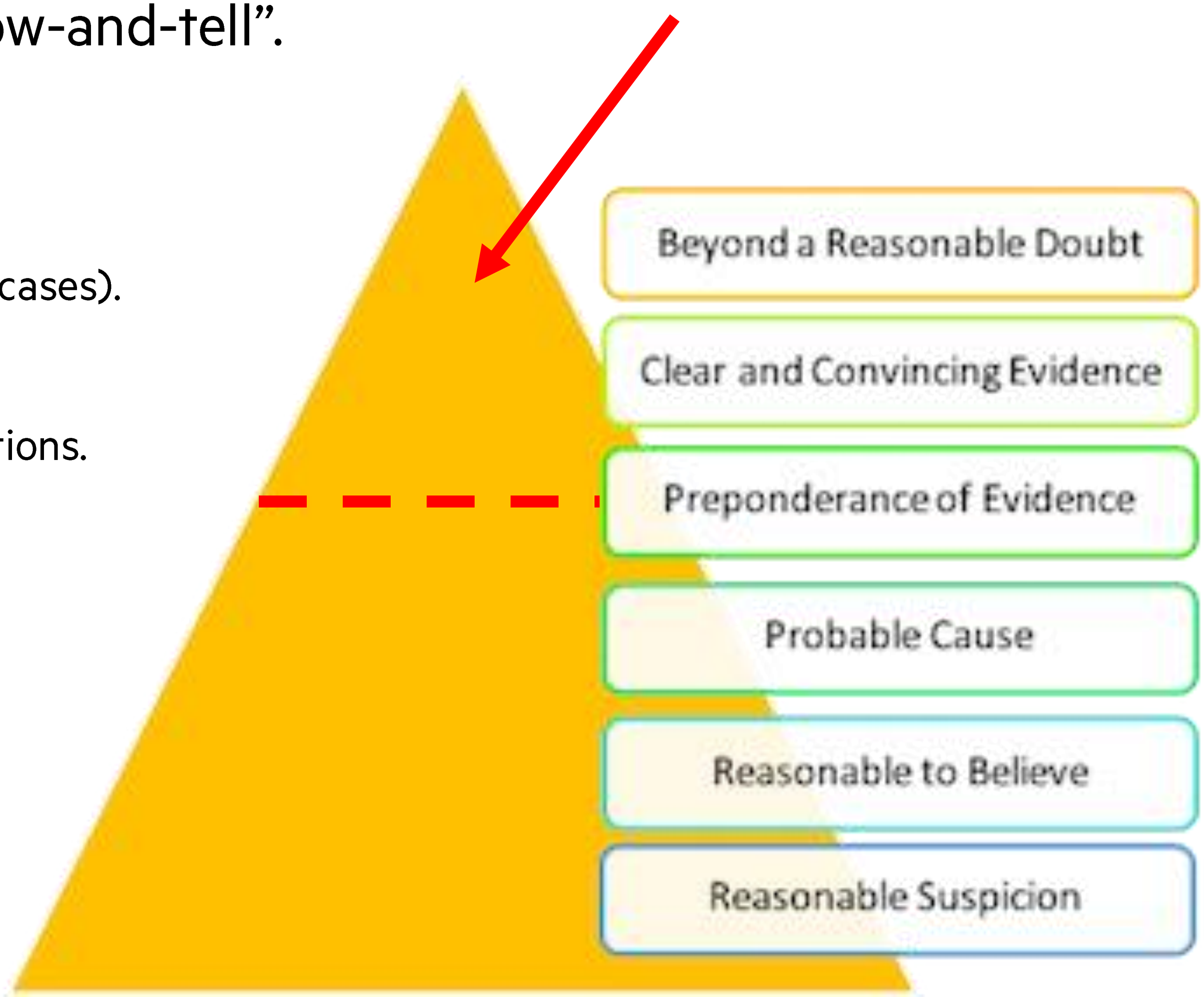
Software engineers
Computational labs
System admins
Gov’t labs

Univ. faculty
Individual scientists
Tutorials
Workshops

Scientists and Computers: Fun Facts

4. Scientists strongly prefer the “show” in “show-and-tell”.

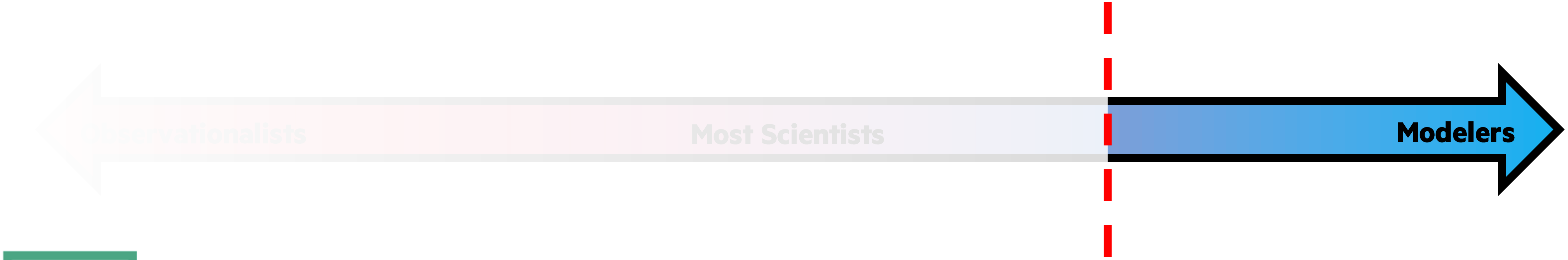
- ***Is it hype? Or is it legit?***
- Easier to see where the solution fits in their own work (e.g. use cases).
- Scientists are naturally skeptical.
- Scientists are trained to sniff out incomplete or inaccurate solutions.
- Burden of proof is squarely on *you*.


Beyond a Reasonable Doubt
Clear and Convincing Evidence
Preponderance of Evidence
Probable Cause
Reasonable to Believe
Reasonable Suspicion

Scientists and Computers: Fun Facts

5. Most scientists are easily waylaid by DIY software setup.

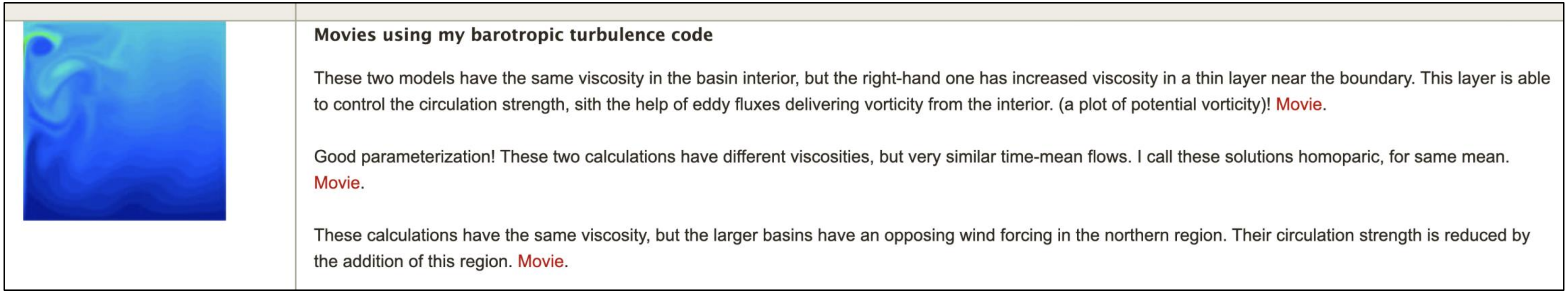
- Things like *compiling* and *linking libraries* can be very foreign concepts.
- Many scientists do not have admin privileges over their clusters.
- Many scientists only have basic familiarity with their clusters.
- Many scientists *do not know what is possible.*
- Many institutions provide slow and ineffective tech support.
    - *Scientists may just move on, rather than wait or wrestle to get it working!*



Scientists and Computers: Fun Facts

6. Scientists develop a strong loyalty to solutions that “just work”.

   - Scientists want to spend time on science, not software.
   - Scientists get excited by solutions that are easy to set up and work well.
   - Scientists will often re-use solutions. A LOT.
   - Scientists will *share* good solutions with other scientists, especially students.



Movies using my barotropic turbulence code

These two models have the same viscosity in the basin interior, but the right-hand one has increased viscosity in a thin layer near the boundary. This layer is able to control the circulation strength, sith the help of eddy fluxes delivering vorticity from the interior. (a plot of potential vorticity)! Movie.

Good parameterization! These two calculations have different viscosities, but very similar time-mean flows. I call these solutions homoparic, for same mean. Movie.

These calculations have the same viscosity, but the larger basins have an opposing wind forcing in the northern region. Their circulation strength is reduced by the addition of this region. Movie.

Working MATLAB code that is *still* hosted on my Ph.D. advisor’s website, nearly 20 years after it was written.

Scientists and Computers: Fun Facts

7. Scientists develop a strong loyalty to languages and programs.

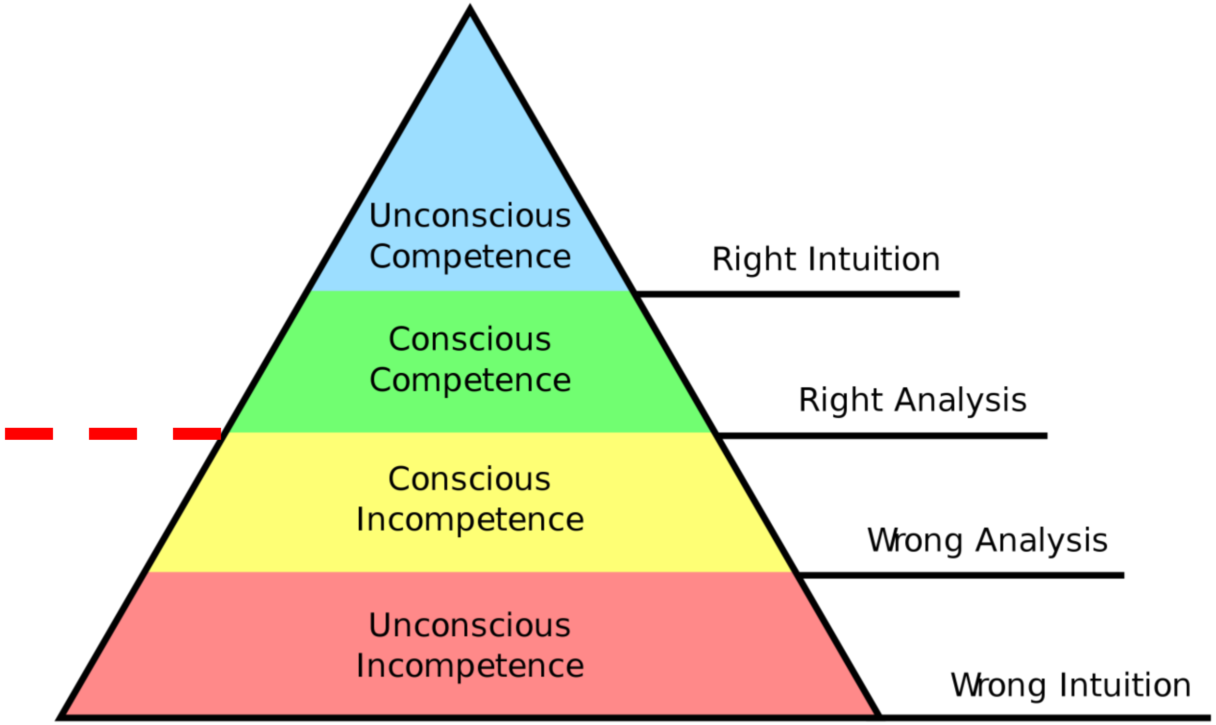
   - Research programs tend to build on themselves / repeat.
   - “I put in all that effort to learn ____”
   - Clear understanding whether the solution will work for the current problem
   - Feeling of “ownership”

WE NEED COBOL PROGRAMMERS FOR OUR MAINFRAME MILLENIUM PROBLEM.
www.unitedmedia.com
S. Adams
IF YOU SEE ANYONE WHO LOOKS LIKE A COBOL PROGRAMMER, LET ME KNOW.
TURN AROUND.
© 1997 United Feature Syndicate, Inc.
11/4/97
ARE YOU A COBOL PROGRAMMER?
NO, BUT I'M OFTEN TOLD I LOOK LIKE ONE.
YOU'RE HIRED.

Scientists and Computers: Fun Facts

8. Scientists develop inertia against learning new solutions, languages, and programs.

- "______ worked before, and it will work again."
- Proficiency takes time. Time is precious.
- There is a competency threshold for performing cutting-edge science.
- Is language _______ REALLY necessary to solve my problem?

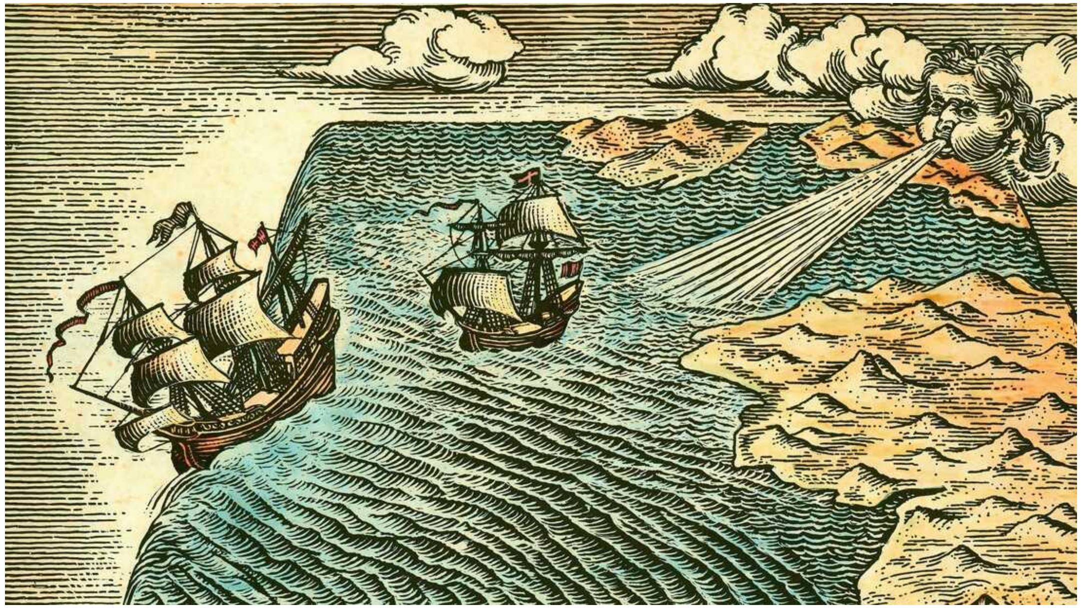


Hierarchy of Competence

Scientists and Computers: Fun Facts

9. Scientists don’t have the bandwidth to seek incremental performance improvements on their own.

    - Incremental improvement may not be worth the time.
    - There is a software engineer down the hall who could do it for me…
    - Lack the time/skill to self-improve.
    - Computer science and technology may be *terra incognita*.

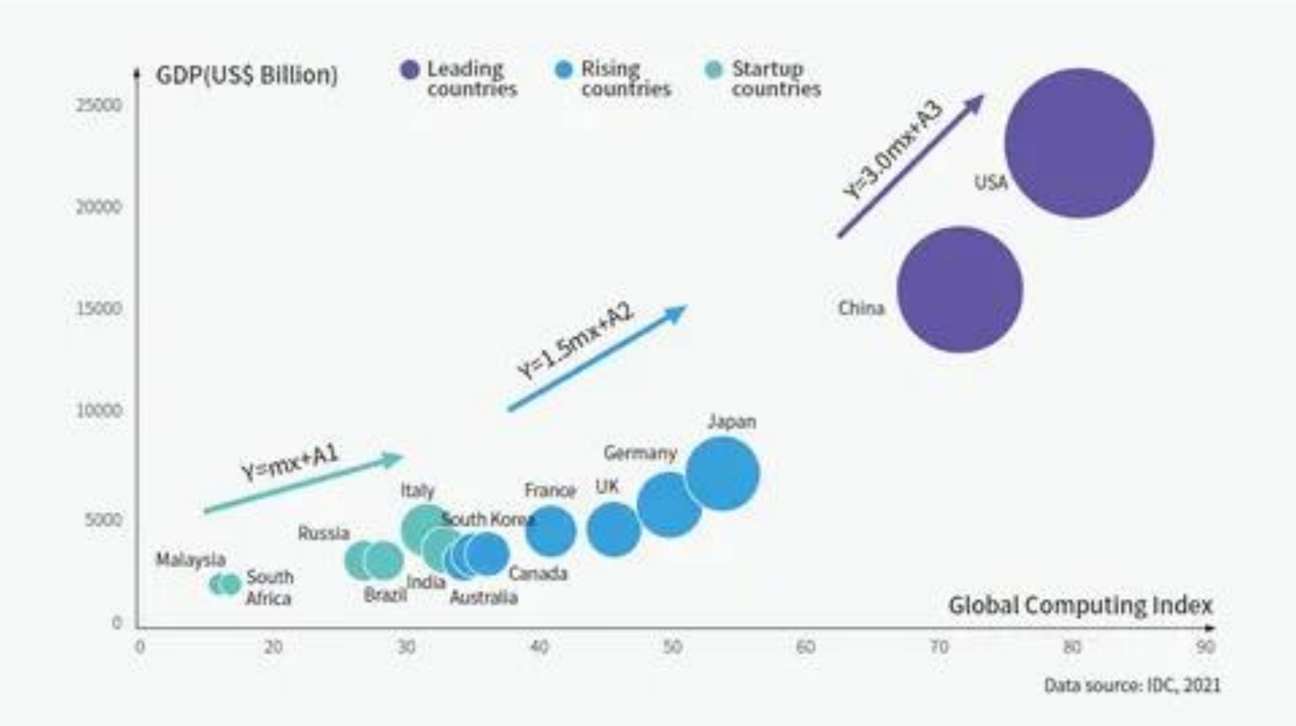


Antar Dayal/Getty Images/Illustration Works

Scientists and Computers: Fun Facts

10. Scientists’ careers are shaped by the computing solutions that are available to them.

- Compute power ⟺ problem size
- Have you spent time at a national lab? Do you know someone there?
- Can you obtain the program / code / solution you need?
- Naturally leads to research and disciplinary silos
  - Fight back with Open Source and democratization!



Regression Analysis of Computing Index and GDP (Graphic: Business Wire)
https://www.businesswire.com/news/home/20220715005001/en/

Scientists and Computers: Fun Facts

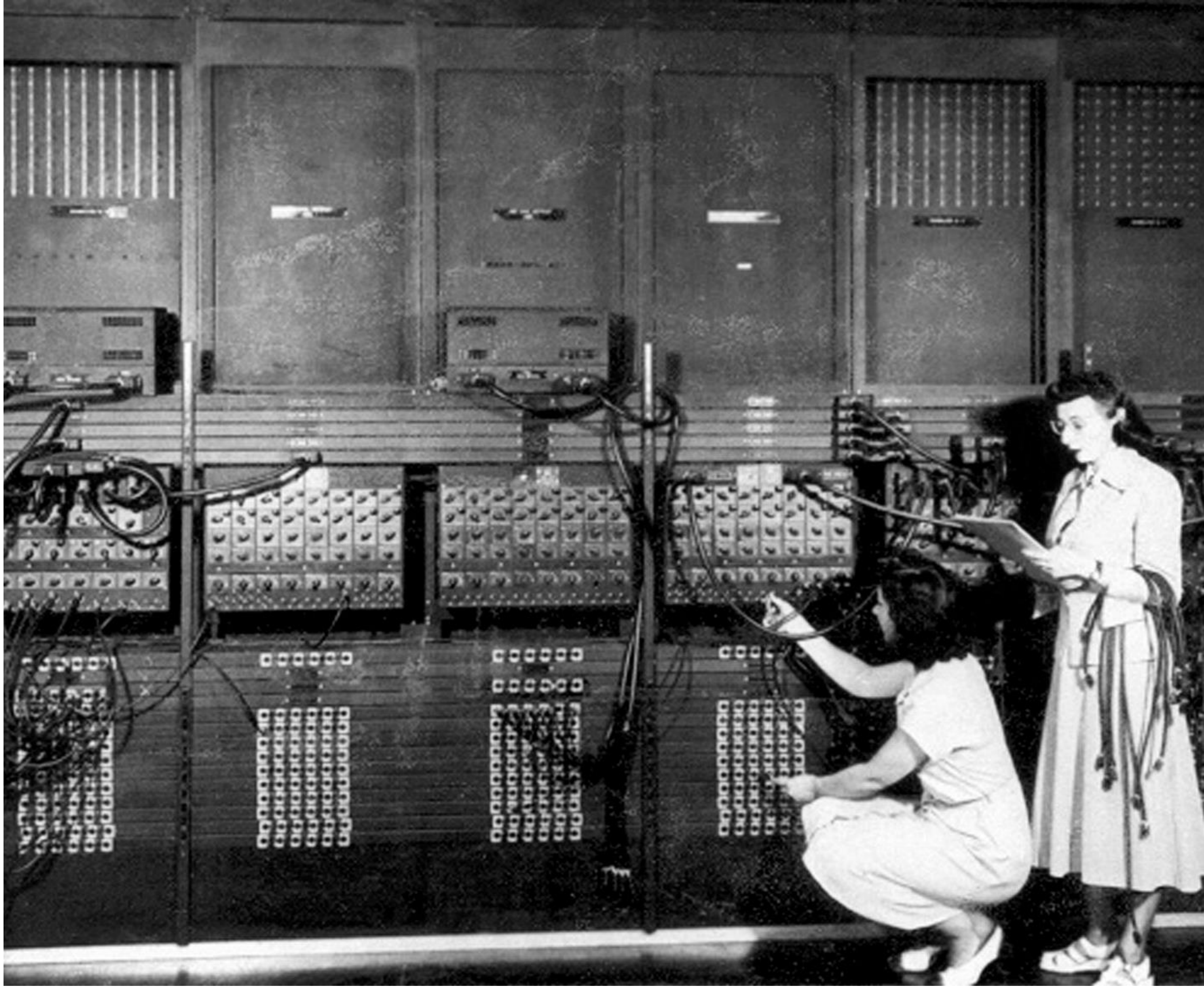
11. Scientists LOVE software engineers.

Because you make everything possible…

HPC and Scientific Computing:
From feudalism to democratization

Historical HPC in the weather domain

- Numerical simulations of weather stretch back to the ENIAC (1950)
  - Even today, some weather/climate models contain lines of code first written in the 1970s
- Many concepts commonplace now in HPC were manual processes and/or were developed hand-in-hand with hardware
- Specialized hardware, skills, and knowledge => Concentration of expertise in silos => Rise of “feudal states” (e.g. national labs)



ENIAC 1952: Getty Research Institute

Artifacts from the feudal era discovered by an amateur code archaeologist (me)

- Strong competition between academic/government fiefdoms meant that advances in knowledge were shared but not the tools
- Postdocs at national labs can make entire careers in academia because of access to code/knowledge
- Siloing means that both the model AND infrastructure to run the model have long lineages
  - Strongly interconnected
  - Backwards compatibility

Getty Images

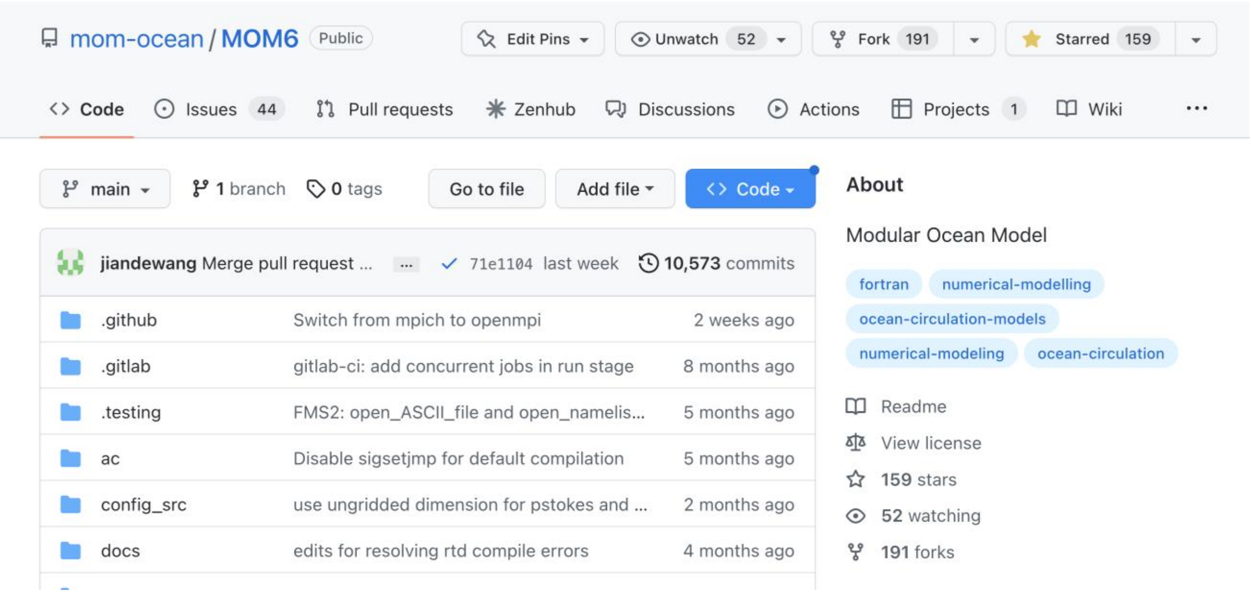
Persistent cultural artifacts from that era

- The good ideas of yesterday form the legends of today
  - Programming “tricks” do not necessarily apply to modern day
- Specialization of knowledge and hardware meant “roll your own” was the only viable way forward
- Reverse card: Are ‘modern’ software languages needed?
  - Do they match needs/requirements?
    - Write once, use forever
    - Long ‘release’ timelines (~5 years)
  - Maybe the right path for ‘new’ simulations
- Legacy doesn’t mean ‘old’ code; it means history and knowledge

Towards a more democratic era of scientific computation

- Key drivers from the feudal era to the democratization
  - Open-source software: reproducibility, skeptical evaluation, and sharing
  - Cheaper/easier HPC hardware: Lower barrier to entry and portability
  - Generational turnover in culture and personnel

- The new reality:
  - *Software engineering is on an equal footing to scientific advancement*
    – Technical debt impedes scientific advancement
  - Community use and criticism of your code drives innovation
  - Modern software engineering principles/frameworks lead to
    – Higher productivity
    – Faster onboarding, easier transfer of knowledge



mom-ocean / MOM6 Public

jiandewang Merge pull request ... 71e1104 last week 10,573 commits

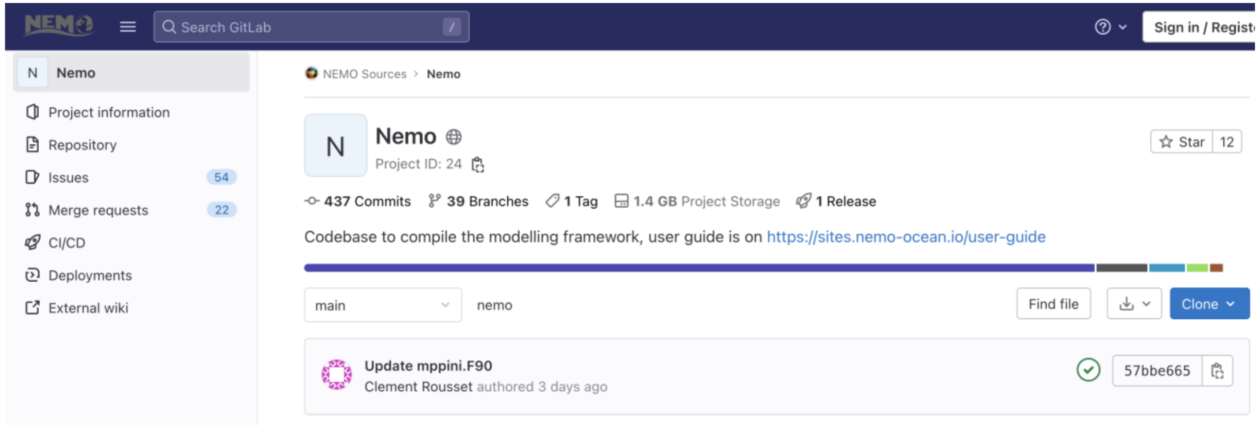
.github	Switch from mpich to openmpi	2 weeks ago
.gitlab	gitlab-ci: add concurrent jobs in run stage	8 months ago
.testing	FMS2: open_ASCII_file and open_namelis...	5 months ago
ac	Disable sigsetjmp for default compilation	5 months ago
config_src	use ungridded dimension for pstokes and ...	2 months ago
docs	edits for resolving rtd compile errors	4 months ago

About

Modular Ocean Model

fortran numerical-modelling ocean-circulation-models numerical-modeling ocean-circulation

Readme
View license
159 stars
52 watching
191 forks



NEMO Sources › Nemo

Nemo

Project ID: 24

437 Commits 39 Branches 1 Tag 1.4 GB Project Storage 1 Release

Codebase to compile the modelling framework, user guide is on https://sites.nemo-ocean.io/user-guide

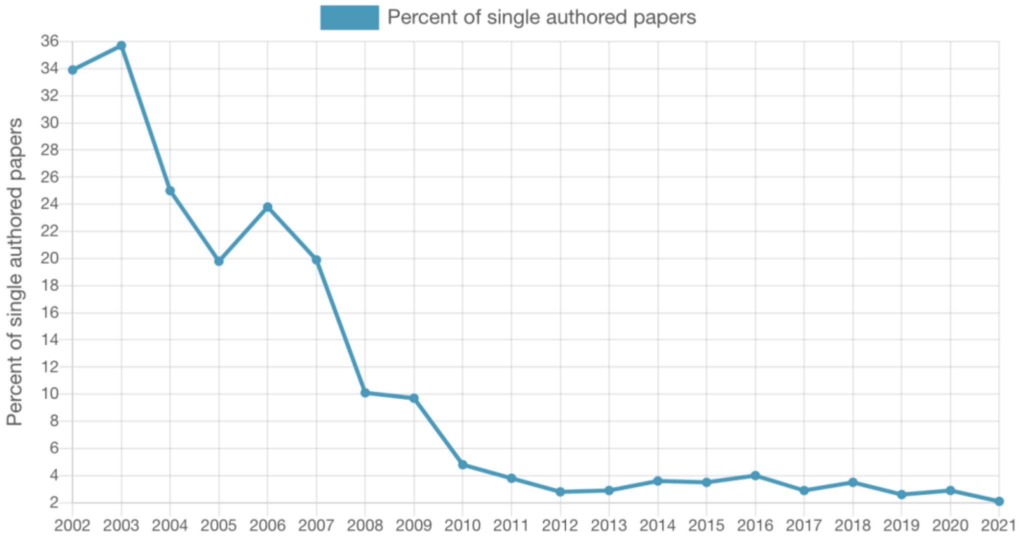
Update mppini.F90
Clement Rousset authored 3 days ago
57bbe665

Making AI/ML available to the masses (of scientists/engineers)

- Present day reality has mismatches between simulation and AI
  - Skillsets/philosophy
    - Scientists are trained to go from fundamental principles to solution
    - ML goes from data-first and develops ad-hoc relationships
  - Hardware:
    "Our scientific simulations have the arithmetic intensity of a potato" – Ocean model developer
    - Is the hardware 'correct' for the problem?

- Continuing scientific advancement requires collaboration amongst scientists of all flavors
  - Too much knowledge and information for any one person (or domain) to know

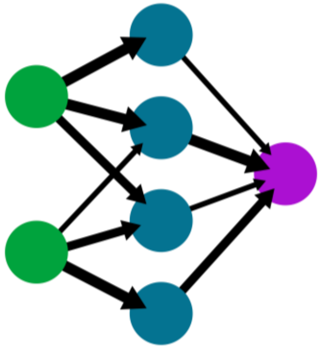
- Open source software can be where cultures meet
  - Provides a 'common language'
  - Community-vetting of tools
  - Lowers barriers to entry



https://quantifyinghealth.com/number-of-authors-of-research-papers/

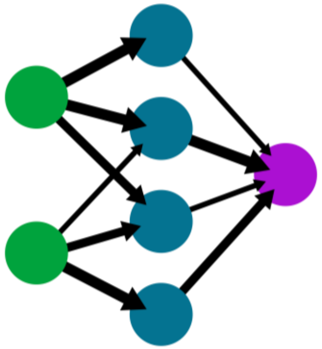
What Computer and Domain Scientists can do together: A case study in climate

The problem of combining AI/ML and climate



	Climate Scientist	Machine Learning Engineer
Knowledge	Physics and mathematics	AI and computer science
Hardware	CPU	GPU
Software Language	Fortran	Python

HPE’s solution for the software/hardware



+ SMARTSIM +

HPE’s open source SmartSim library bridges the divide by providing:

- Scalable database for storing ML models and data
  - Support for GPU/CPU workloads
- Inference “engine” to do ML prediction
- Native database communication clients in C/C++/Fortran/Python with minimal changes to simulation code
- **Enables calling ML models for training and inference in legacy code**

The start of a solution: the People part

Scott and NCAR Scientists
"We know climate"

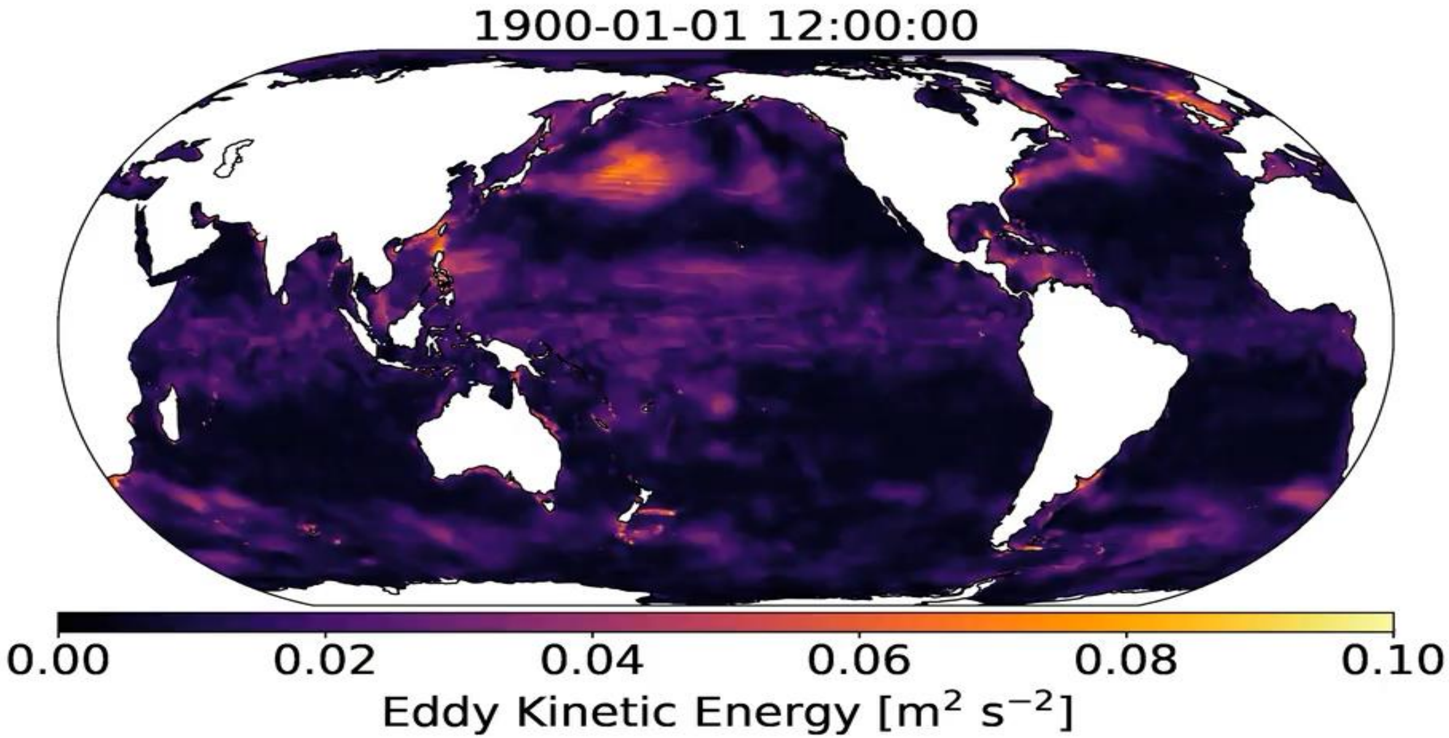
Andrew
"I know a little about both"

SMARTSIM

HPE Engineers
"We know machine learning"

- Life Lesson 1: Find the people who will take the time to learn from each other
- Life Lesson 2: Trust each other's expertise to cover each other's knowledge gaps

It works! First demonstration of online prediction in a realistic ocean simulation



- 970 billion inferences over 10 simulation years
- ~12,000 CPU cores but only 16 GPUs (efficient use of expensive resources!)
- All necessary code and training scripts were released as open source
- Available and being used by MOM6 users

Summary and next steps

Ocean/Climate Modelling

- SmartSim is now an ‘official’ solution for ML inside MOM6
  - Compatible open-source license was critical for acceptance into trunk
- Gaining interest within the NEMO ocean model community
  - Solution is open-source
  - Scientific demonstration is freely available

Larger-scale value to commercial sector

“I didn’t realize how quickly we’d be able to start using ML”
- Anonymous Principal Engineer

- Removes the technical barriers to creating simulation/AI applications
- Users can spend more time experimenting instead of creating infrastructure
- Allows users to only use GPU resources as needed (cheaper overall to prototype and run)
- Promotes thinking about simulations as part of a larger application

Chapel and Arkouda:
Exemplary models of democratized HPC

What is Chapel?

Chapel: A modern parallel programming language

- portable & scalable
- open-source & collaborative

Goals:

- Support general parallel programming
- Make parallel programming at scale far more productive

Chapel's Multiresolution Philosophy

1. Users should be able to program at high levels of abstraction and get good performance**

```
Dst = Src[Inds];     // whole-array index gather
```

2. Yet, when more control / better performance is needed, they can drop to lower levels…

```
forall (d, i) in zip(Dst, Inds) do     // parallel loop-based index gather
  d = Src[i];
```

..and even lower levels, as necessary…

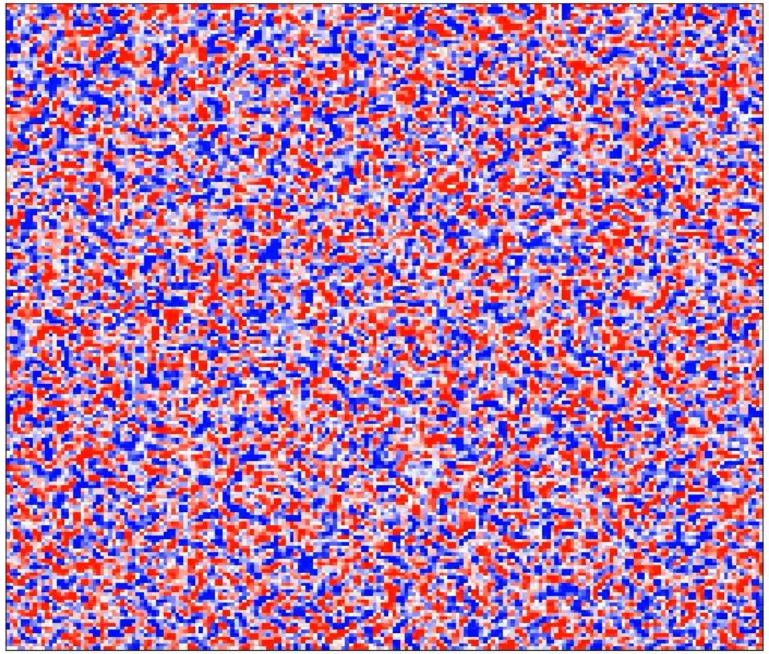
```
coforall loc in Dst.targetLocales do     // explicit SPMD-style index gather
  on loc do
    forall i in Dst.localSubdomain do
      Dst.localAccess[i] = Src[Inds.localAccess[i]];
```

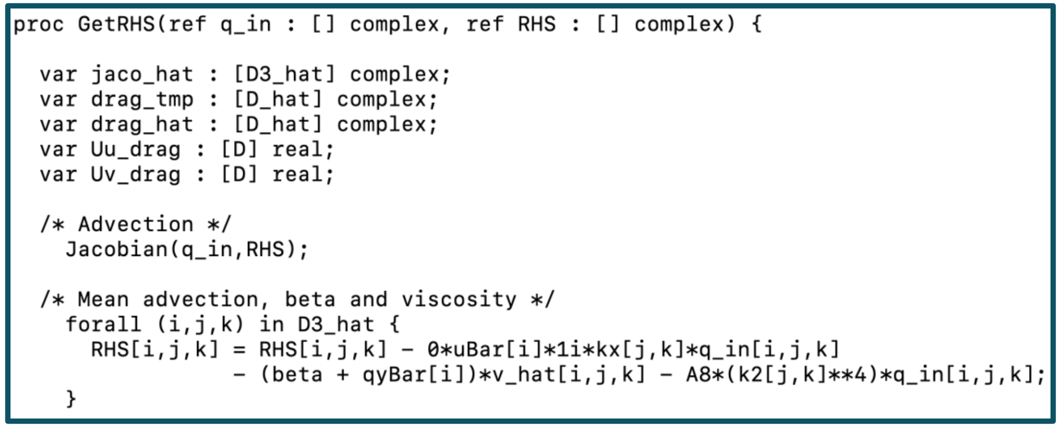
..where "calling out to C / CUDA / MPI / etc." is effectively the lowest level

3. Chapel builds its higher-level abstractions in terms of the lower-level ones to guarantee compatibility

** - Distributed computing for the masses!

Chapel is amazing!





```
proc GetRHS(ref q_in : [] complex, ref RHS : [] complex) {

  var jaco_hat : [D3_hat] complex;
  var drag_tmp : [D_hat] complex;
  var drag_hat : [D_hat] complex;
  var Uu_drag : [D] real;
  var Uv_drag : [D] real;

  /* Advection */
    Jacobian(q_in,RHS);

  /* Mean advection, beta and viscosity */
    forall (i,j,k) in D3_hat {
      RHS[i,j,k] = RHS[i,j,k] - 0*uBar[i]*1i*kx[j,k]*q_in[i,j,k]
                 - (beta + qyBar[i])*v_hat[i,j,k] - A8*(k2[j,k]**4)*q_in[i,j,k];
    }
```

… but models like this are “set and forget”…

… and the VAST majority of scientists do analysis, not modeling…

OCEAN SCIENCES MEETING | FEBRUARY 18-23 2024
NEW ORLEANS, LA & ONLINE EVERYWHERE

= ~5,000 attendees

COMMODORE workshop
Community for the numerical modeling of the global, regional and coastal ocean

= ~20 attendees

… and most analysis workflows are *exploratory, spontaneous, bespoke, and evolve constantly…*

Analysis is where all the action is ! How do we serve those potential customers?

Enter Arkouda

Motivation: Say you've got...

...a bunch of Python programmers

...HPC-scale data science problems to solve

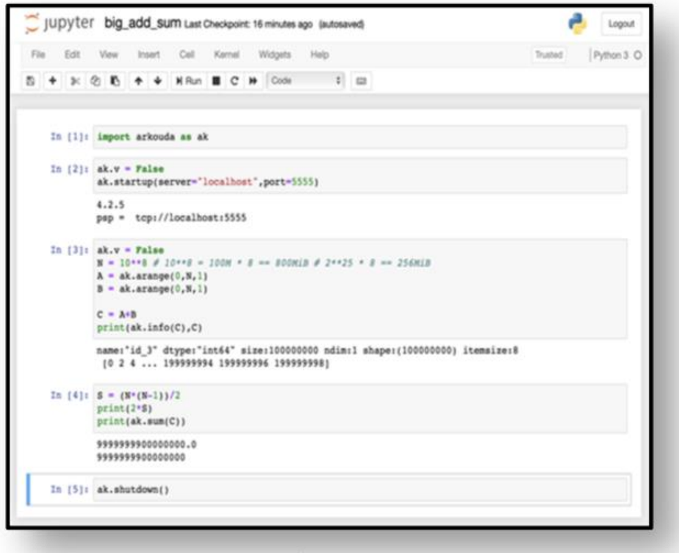
...access to HPC systems

https://www.cscs.ch/computers/piz-daint/

How will you leverage your Python programmers to get your work done?

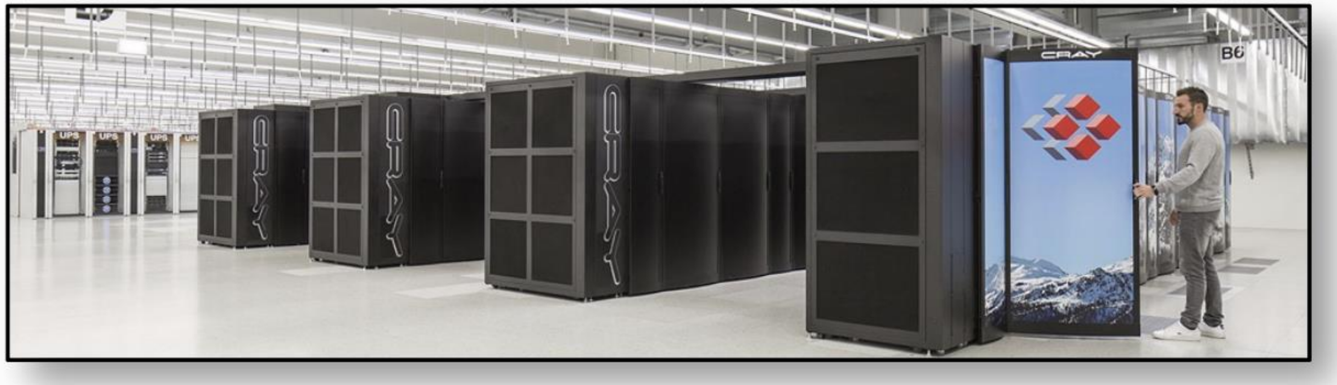
Arkouda's Approach

Arkouda Client
(written in Python)



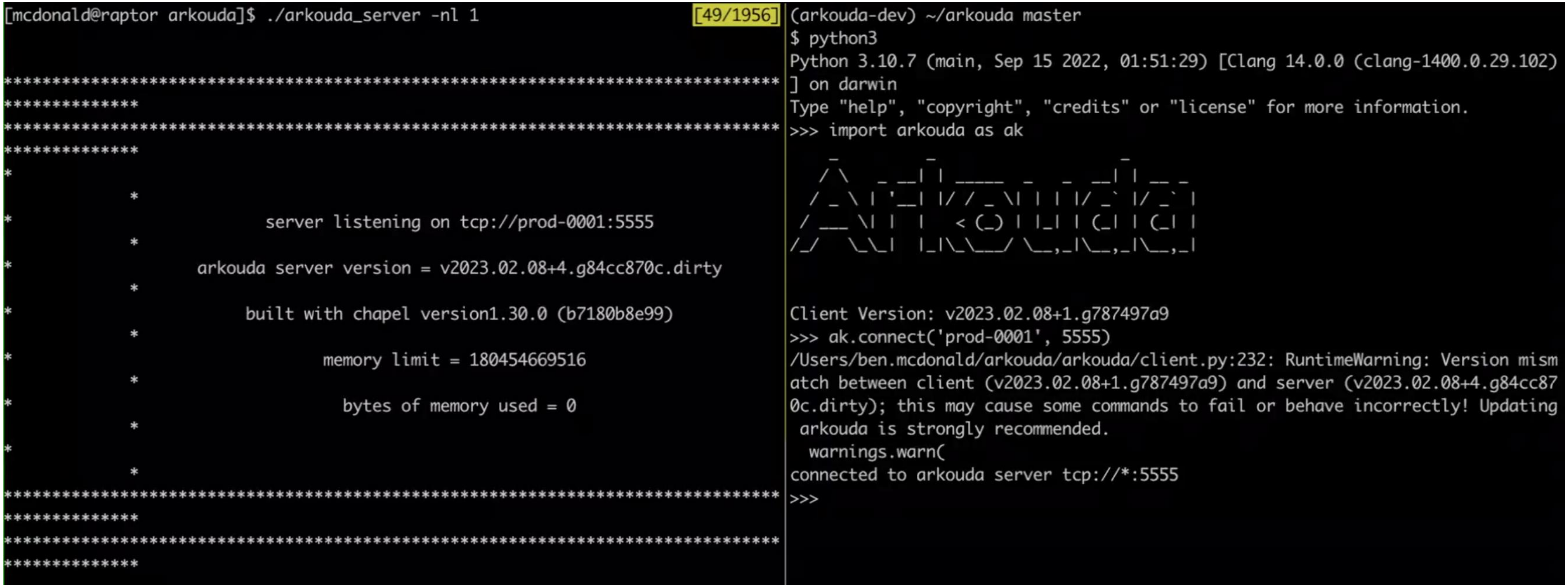
```
In [1]: import arkouda as ak

In [2]: ak.v = False
        ak.startup(server="localhost",port=5555)

        4.2.5
        psp =  tcp://localhost:5555

In [3]: ak.v = False
        N = 10**8 # 10**8 = 100M * 8 == 800MiB # 2**25 * 8 == 256MiB
        A = ak.arange(0,N,1)
        B = ak.arange(0,N,1)

        C = A+B
        print(ak.info(C),C)

        name:"id_3" dtype:"int64" size:100000000 ndim:1 shape:(100000000) itemsize:8
         [0 2 4 ... 199999994 199999996 199999998]

In [4]: S = (N*(N-1))/2
        print(2*S)
        print(ak.sum(C))

        9999999900000000.0
        9999999900000000

In [5]: ak.shutdown()
```

Arkouda Server (written in Chapel)



Writes Python code in Jupyter
Invoking NumPy/Pandas ops

HPC-scale analysis from the comfort of your own home



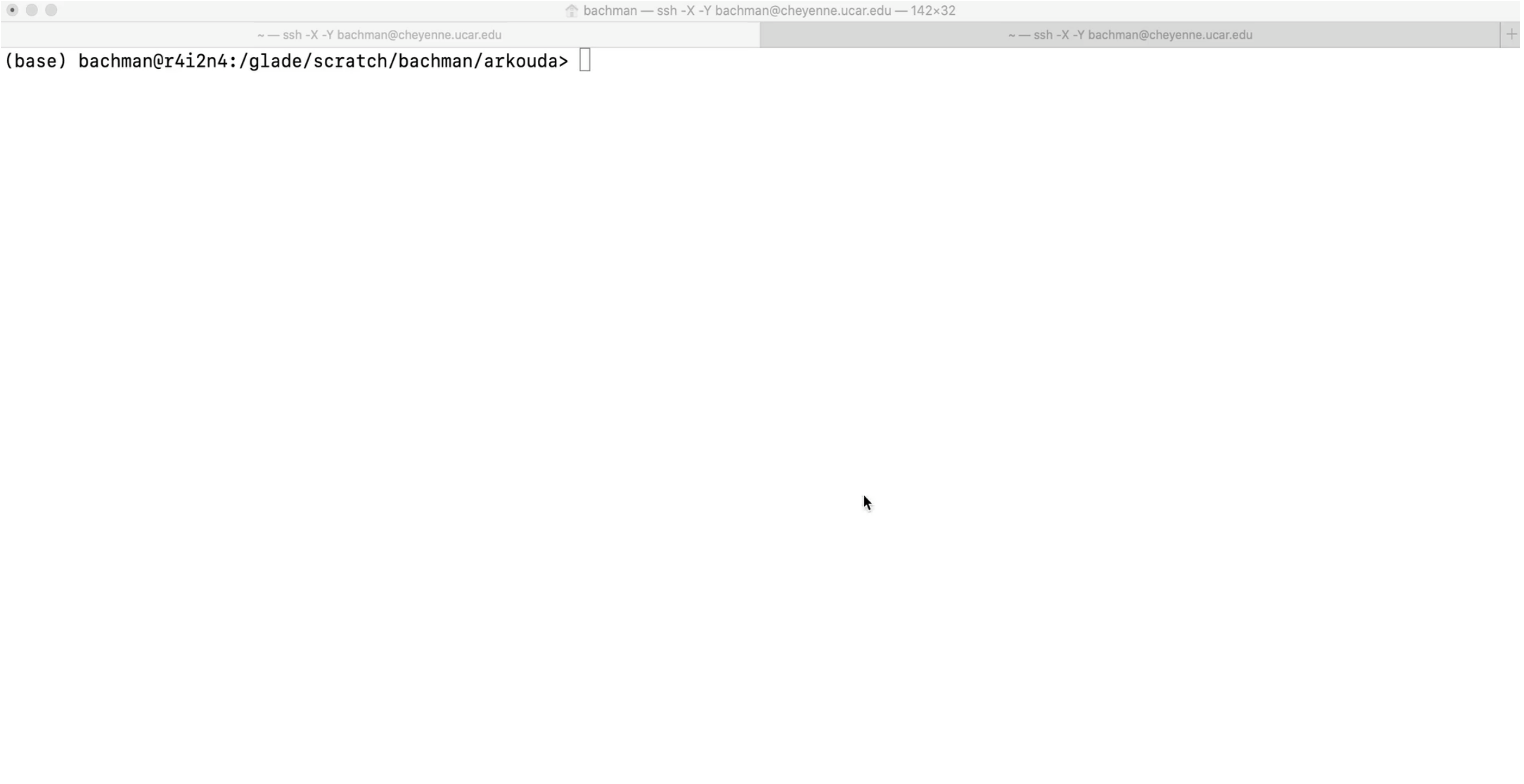
```
[mcdonald@raptor arkouda]$ ./arkouda_server -nl 1                      [49/1956]


*******************************************************************************
**************
*******************************************************************************
**************
*
             *
*                          server listening on tcp://prod-0001:5555
             *
*                   arkouda server version = v2023.02.08+4.g84cc870c.dirty
             *
*                        built with chapel version1.30.0 (b7180b8e99)
             *
*                                memory limit = 180454669516
             *
*                                  bytes of memory used = 0
             *
*
             *
*******************************************************************************
**************
*******************************************************************************
**************
```

```
(arkouda-dev) ~/arkouda master
$ python3
Python 3.10.7 (main, Sep 15 2022, 01:51:29) [Clang 14.0.0 (clang-1400.0.29.102)
] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import arkouda as ak
    _         _                   _
   / \   _ __| | _____  _   _  __| | __ _
  / _ \ | '__| |/ / _ \| | | |/ _` |/ _` |
 / ___ \| |  |   < (_) | |_| | (_| | (_| |
/_/   \_\_|  |_|\_\___/ \__,_|\__,_|\__,_|


Client Version: v2023.02.08+1.g787497a9
>>> ak.connect('prod-0001', 5555)
/Users/ben.mcdonald/arkouda/arkouda/client.py:232: RuntimeWarning: Version mism
atch between client (v2023.02.08+1.g787497a9) and server (v2023.02.08+4.g84cc87
0c.dirty); this may cause some commands to fail or behave incorrectly! Updating
 arkouda is strongly recommended.
  warnings.warn(
connected to arkouda server tcp://*:5555
>>>
```

Courtesy: B. McDonald

This *is* democratized HPC!

HPC-scale analysis from the comfort of your own home


bachman — ssh -X -Y bachman@cheyenne.ucar.edu — 142×32
~ — ssh -X -Y bachman@cheyenne.ucar.edu
~ — ssh -X -Y bachman@cheyenne.ucar.edu

```
(base) bachman@r4i2n4:/glade/scratch/bachman/arkouda> 
```

This *is* democratized HPC!

Summary

- SmartSim allows users to build complex AI and simulation applications today
  - “It just works” even if you don’t understand the entire tech stack

- Chapel helps to bring HPC to the masses
  - Arkouda brings Chapel to the masses via Python

- **Democratization of HPC** is more than just “sharing is caring”
  - Fundamental for progress!

- Academia is more open and collaborative than industry tends to be
  - Secrecy is frowned upon => Anything closed source is viewed with suspicion

- Sharing and communication are crucial for scientists to make advances
  - OSS is the mechanism to make that possible

Thank you

Andrew.Shao@hpe.com
Scott.Bachman@hpe.com