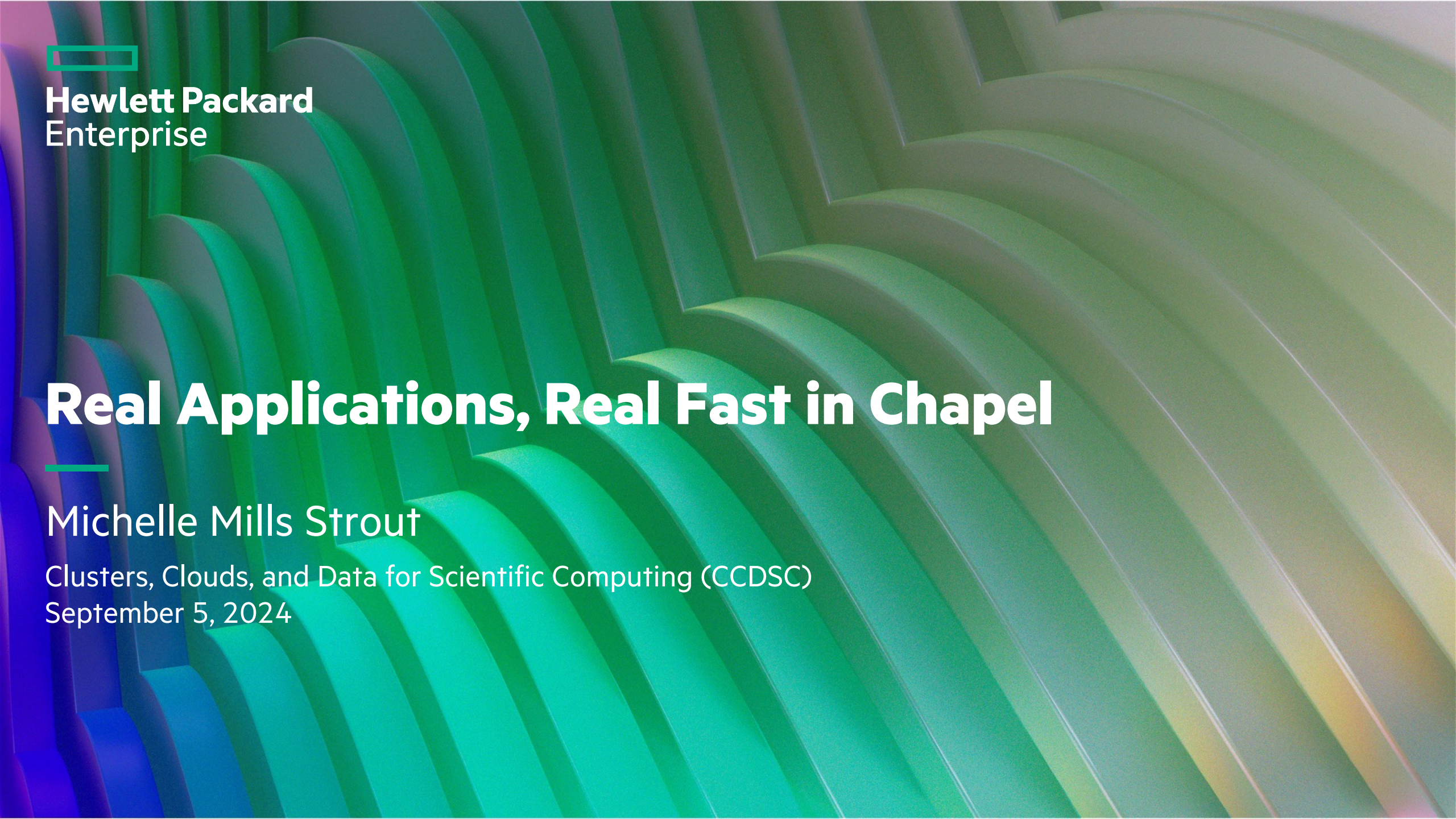


Real Applications, Real Fast in Chapel

Michelle Mills Strout

Clusters, Clouds, and Data for Scientific Computing (CCDSC)
September 5, 2024

What Scientific Computing Practitioners Care About

- Cost of developing, extending, and maintaining parallel and distributed applications
  - Clouds provide a possible answer for how quickly they can scale a computation to massive datasets

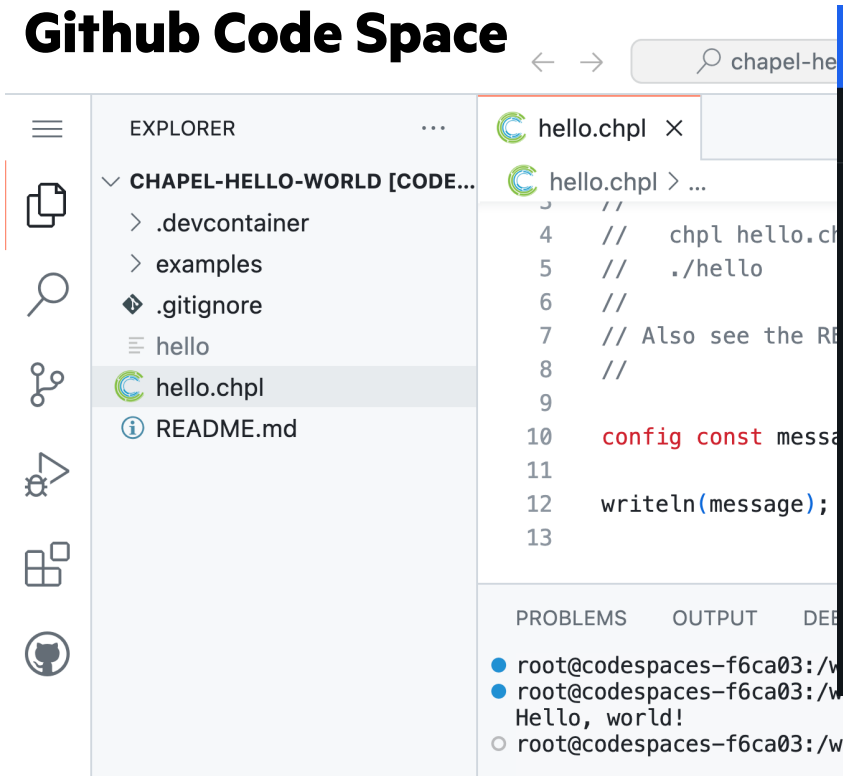
- Performance, Scalability, and Performance Portability are just one of many Metrics
  - Time to try out a new technology
  - Time to first usable application that solves their problem
  - Time to tune application for performance, scalability, and performance portability
  - Time to evolve the application for scientific discovery
  - Ease of use by collaborators
  - Time to maintain application

- Let's look at how the Chapel Parallel Programming Language does with these metrics

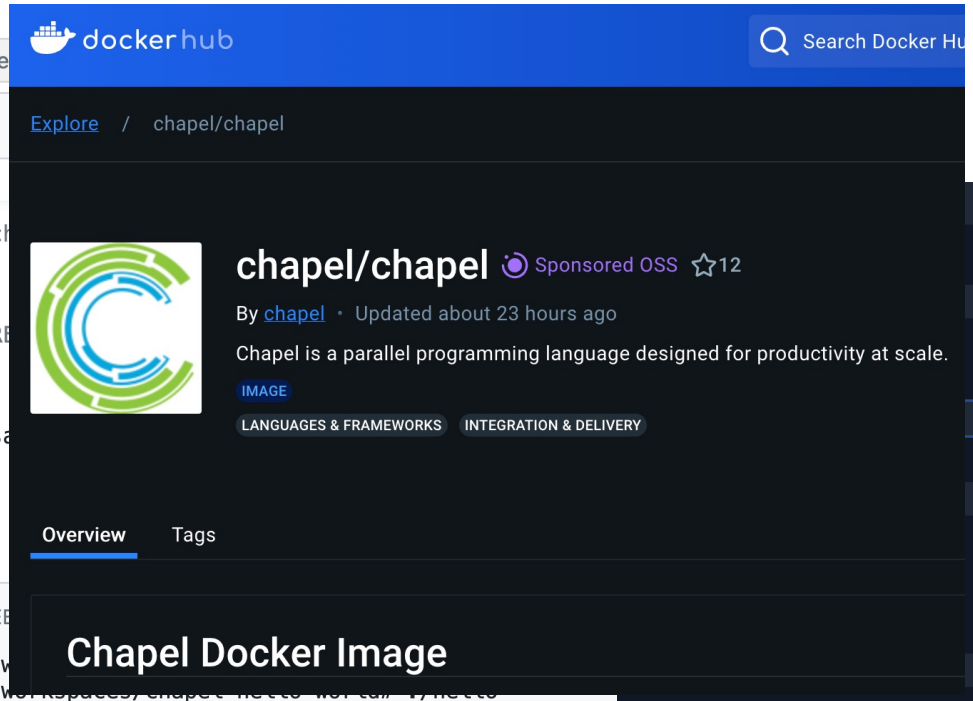
Time to Try Out a New Technology

- How difficult is it to install it?
- Are there smallish examples doing operations the scientists know they will need?

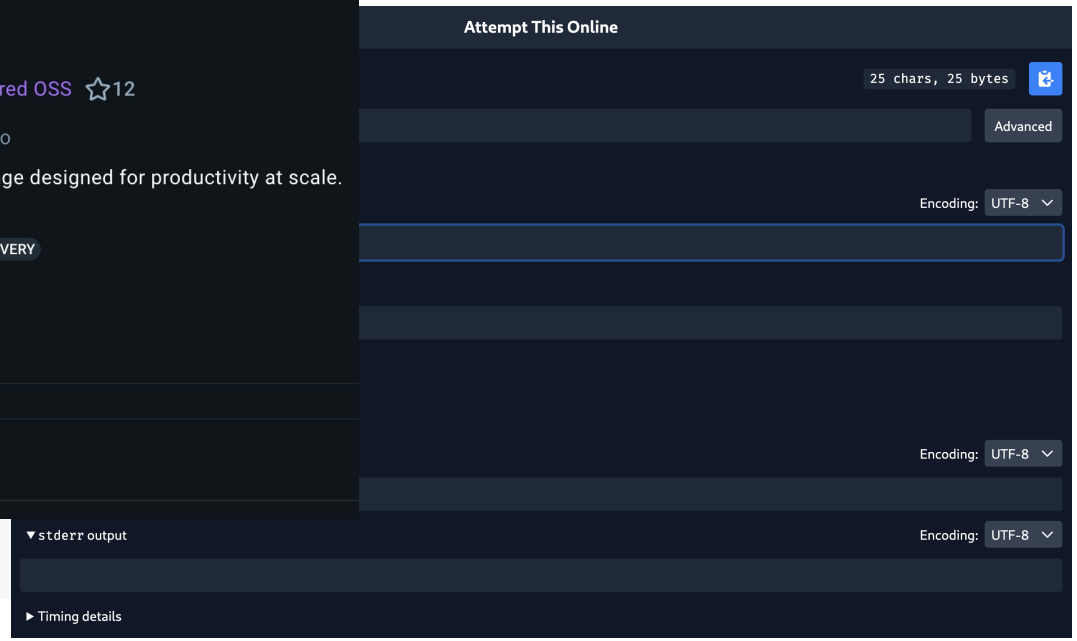
Github Code Space



Docker Containers



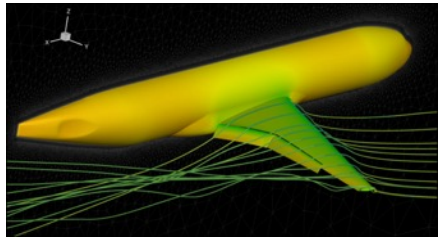
Attempt This Online



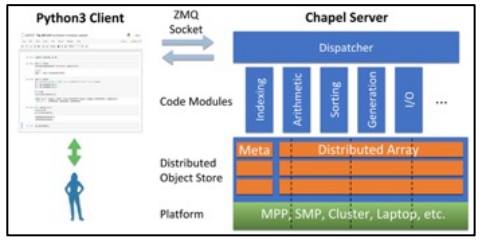
Time to First Usable Application

- How long does it take a new Chapel user to be productive?
  - Many examples in the 3-6 month range

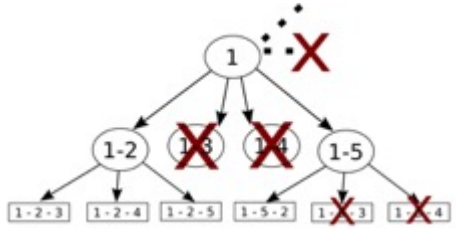
Distributed Applications in Many Domains are Written in Chapel



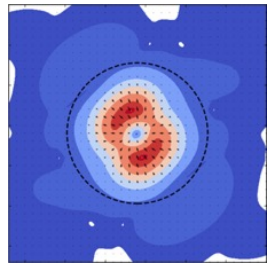
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



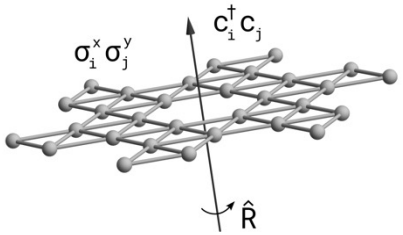
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



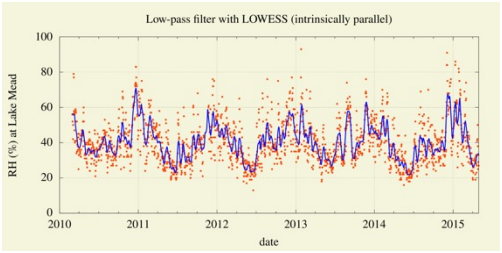
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



ChplUltra: Simulating Ultralight Dark Matter
Nikhil Padmanabhan, J. Luna Zagorac, et al.
Yale University et al.

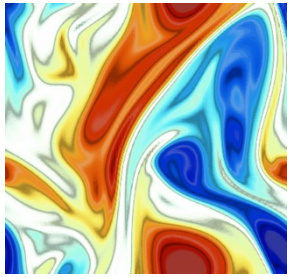


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

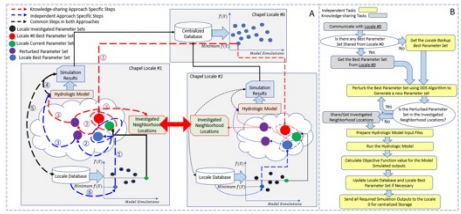


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

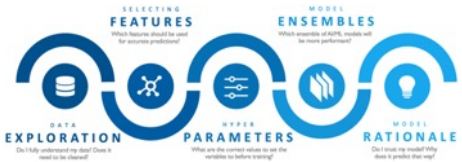
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



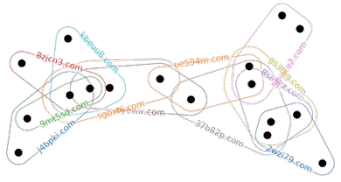
ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.



Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE



CHGL: Chapel Hypergraph Library
Louis Jenkins, Cliff Joslyn, Jesun Firoz, et al.
PNNL

?

Your Application Here?

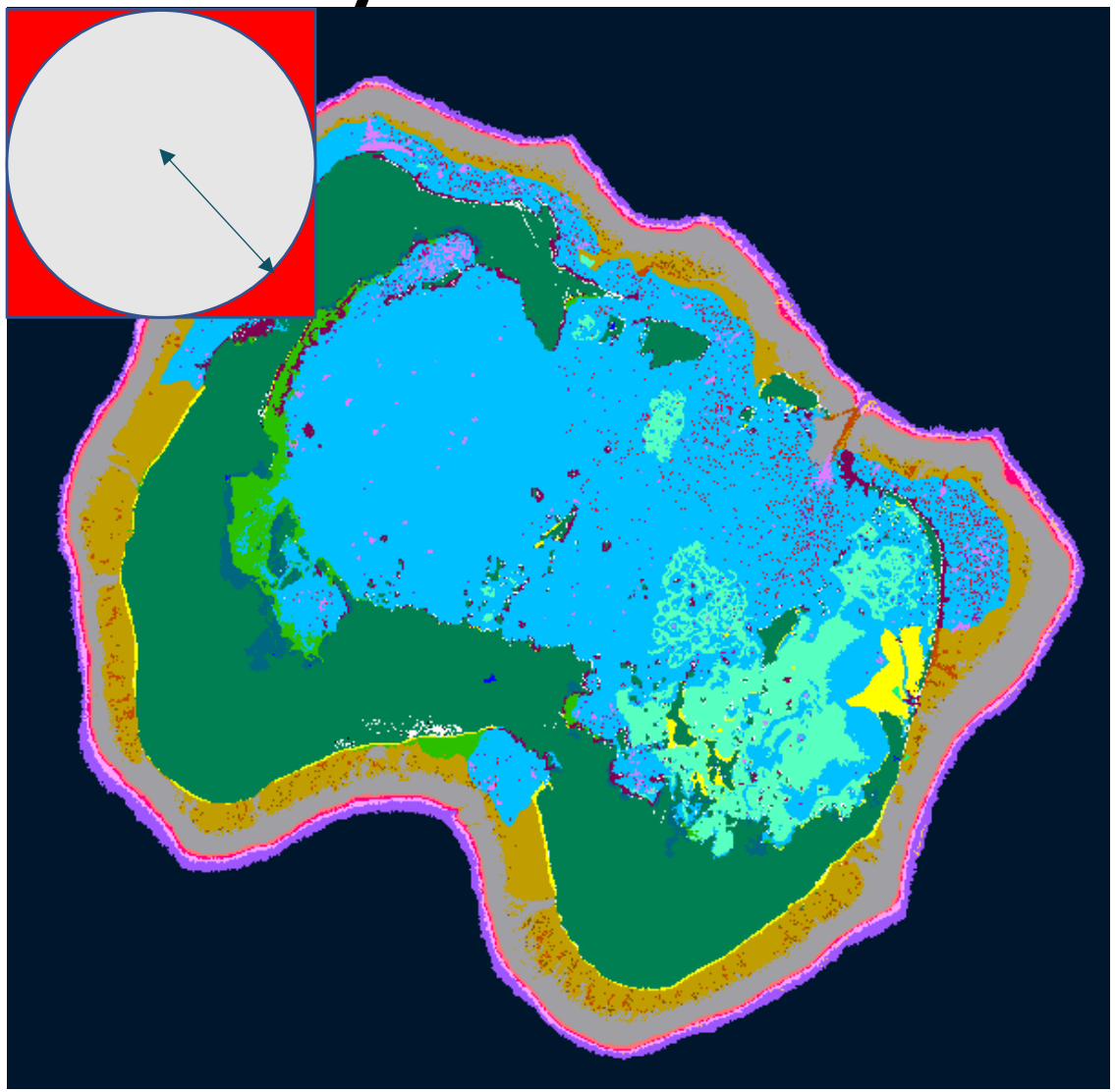
(images provided by their respective teams and used with permission)

Institutions and Application Domains Using Chapel

Institutions	Application Domains
École Polytechnique Montréal in Canada	3D Unstructured CFD for Airplane simulation, CHAMPS
U.S. Govt	Arkouda: Exploratory Data Analysis at Scale
New Jersey Institute of Technology in USA	Distributed Graph Analytics in Arachne/Arkouda
INRIA in France and IMEC in Belgium	Branch and Bound Optimization (e.g., N-Queens), ChOP
Yale University in USA	Distributed FFTs
Radboud University in The Netherlands	Quantum Simulations
The Federal University of Paraná in Brazil	Environmental Engineering
University of Colorado, Boulder in USA	Structured CFD for climate science
University of Guelph in Canada	Hydrological model
PNNL in USA	Hypergraph Library
Cray/AI at HPE	Hyper Parameter Optimization
Coral Reef Alliance	Image Analysis

Use Case: Image Processing for Coral Reef Dissimilarity

- **Analyzing images for coral reef diversity**
  - Important for prioritizing interventions
- **Algorithm implemented productively**
  - Add up weighted values of all points in a neighborhood, i.e., convolution over image
  - Developed by Scott Bachman, NCAR scientist who is a visiting scholar on the Chapel team
  - Scott started learning Chapel in Sept 2022, started Coral Reef app in Dec 2022, already had collaborators presenting results in Feb 2023
  - In July with ~5 lines changed, ran on a GPU
- **Performance**
  - Less than 300 lines of Chapel code scales out to 100s of processors on Cheyenne (NCAR)
  - Full maps calculated in *seconds*, rather than days



Arkouda is the word for "bear" in Greek

ARKOUDA

Arkouda is a framework for interactive, high performance data analytics

- Massive scales = dozens of terabytes
- Interactive rates = operations that run within the human thought loop (i.e., seconds to minutes)
- User observation
  - *No other tool provides Exploratory Data Analysis (EDA) at these scales*
- Extensible in Python and in Chapel
- Open-source: https://github.com/Bears-R-Us/arkouda

ARKOUDA: A CASE STUDY FOR PRODUCTIVITY

Oct 2018: Mike Merrill and Bill Reus took a Chapel tutorial

Dec 2018: Mike reaches out to mention his intent to start building Arkouda over the holidays

May 2019: first-draft counting sorts written and tuned

June 2019: Mike gives first public talk about Arkouda at CHIUW

Sept 2019: looked at NESL LSD radix sort and ~4 hours later had a ~100-line scalable sort

- achieved 80 GiB/s on 512 nodes of Cray XC

Nov 2019: changed ~12 lines of sort code to aggregate small messages

- 40% improvement on Cray XC, ~1000x improvement on InfiniBand

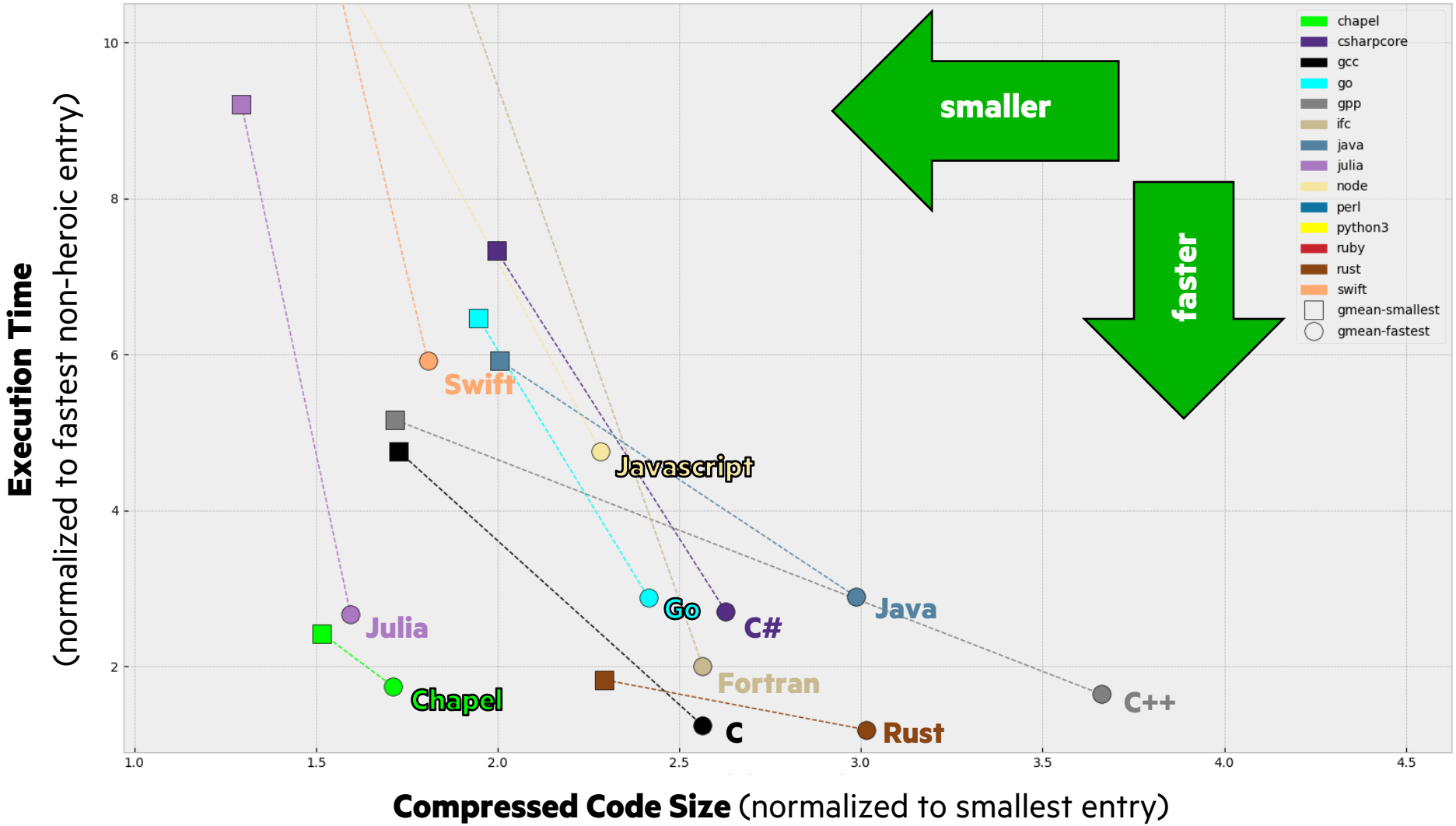
Nov 2019: Mike gives talk about Arkouda at PAW-ATM (SC19)

Time to Tune Applications for Performance, Scalability, and Perf Portability

- Do applications written using Chapel perform well?
- Do they scale?
- How portable is Chapel and its performance?

On Node Chapel Performance and Source Code Size are Fantastic!

CLBG Summary, Aug 19, 2024 (selected languages, no heroic versions)



Arkouda Argsort Performance

HPE Apollo (May 2021)

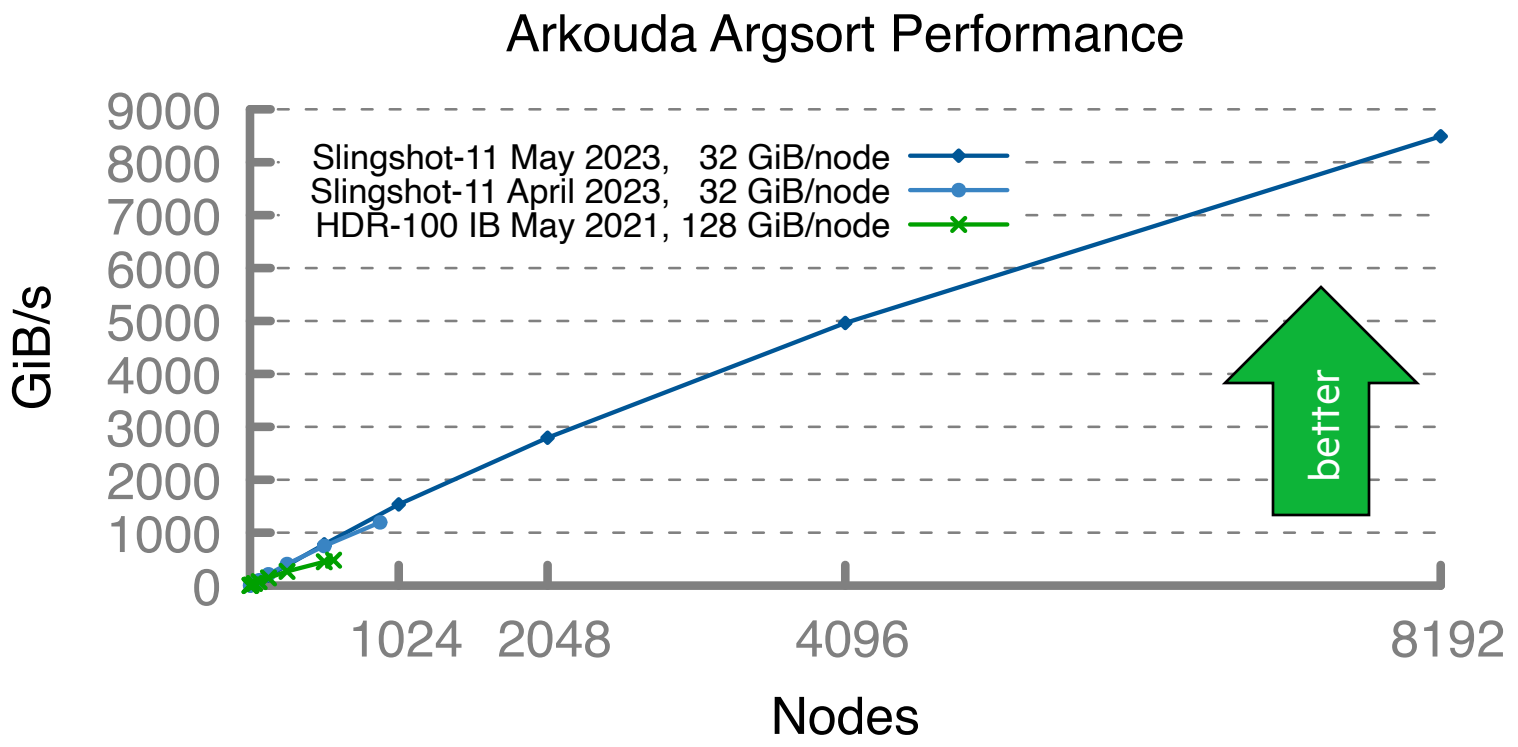
- HDR-100 Infiniband network (100 Gb/s)
- 576 compute nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)

HPE Cray EX (April 2023)

- Slingshot-11 network (200 Gb/s)
- 896 compute nodes
- 28 TiB of 8-byte values
- ~1200 GiB/s (~24 seconds)

HPE Cray EX (May 2023)

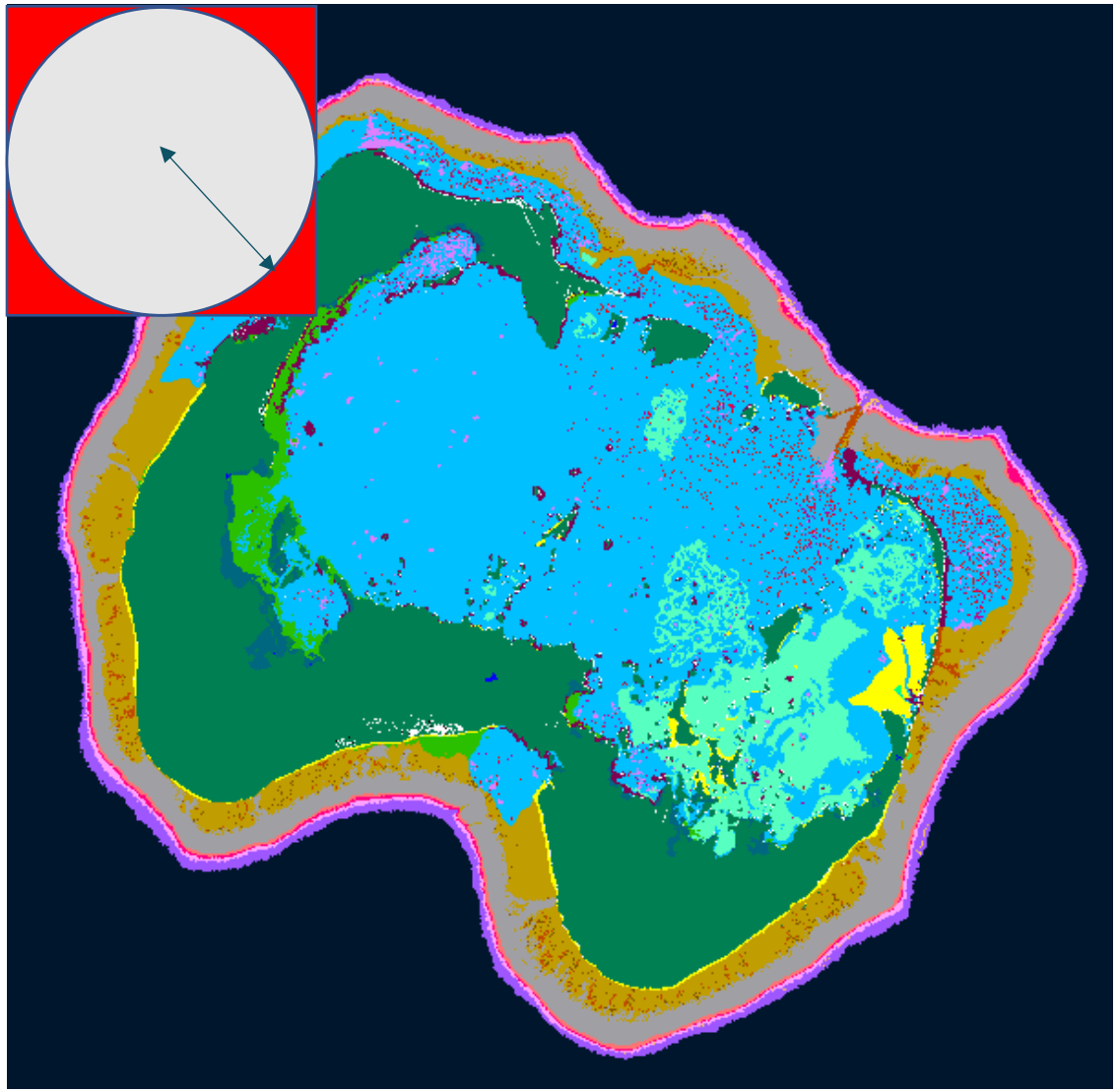
- Slingshot-11 network (200 Gb/s)
- 8192 compute nodes
- 256 TiB of 8-byte values
- ~8500 GiB/s (~31 seconds)



A notable performance achievement in ~100 lines of Chapel

Performance Portability: Image Processing for Coral Reef Dissimilarity

- **Analyzing images for coral reef diversity**
  - Important for prioritizing interventions
- **Algorithm implemented productively**
  - Add up weighted values of all points in a neighborhood, i.e., convolution over image
  - Developed by Scott Bachman, NCAR scientist who is a visiting scholar on the Chapel team
  - Scott started learning Chapel in Sept 2022, started Coral Reef app in Dec 2022, already had collaborators presenting results in Feb 2023
  - In July with ~5 lines changed, ran on a GPU
- **Performance**
  - Less than 300 lines of Chapel code scales out to 100s of processors on Cheyenne (NCAR)
  - Full maps calculated in *seconds*, rather than days



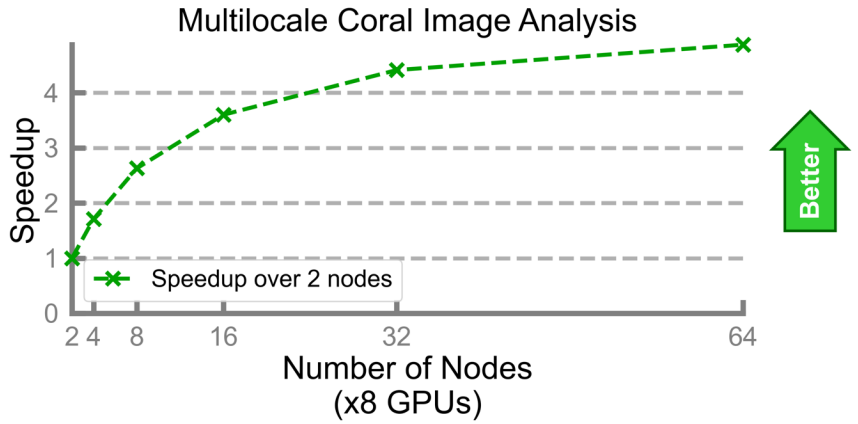
GPU Processing for Coral Reef Spectral Biodiversity

```
proc convolve(InputArr, OutputArr) {   // 3D Inp
 foreach ... {
  tonOfMath();
 }
}
proc main() {
 var InputArr: ...;
 var OutputArr: ...;

 coforall loc in Locales do on loc {      // u
  coforall gpu in here.gpus do on gpu {   // u
    var GpuInputArr = InputArr[...];
    var GpuOutputArr: ...;
    convolve(GpuInputArr, GpuOutputArr);
    OutputArr[...] = GpuOutputArr;
}}}
```

Runs on multiple nodes on Frontier!

- 5x improvement going from 2 to 64 nodes
  - (from 16 to 512 GPUs)
- Straightforward code changes:
  - from sequential Chapel code
  - to GPU-enabled one
  - to multi-node, multi-GPU, multi-thread



Other GPU-enabled Chapel Applications and Performance

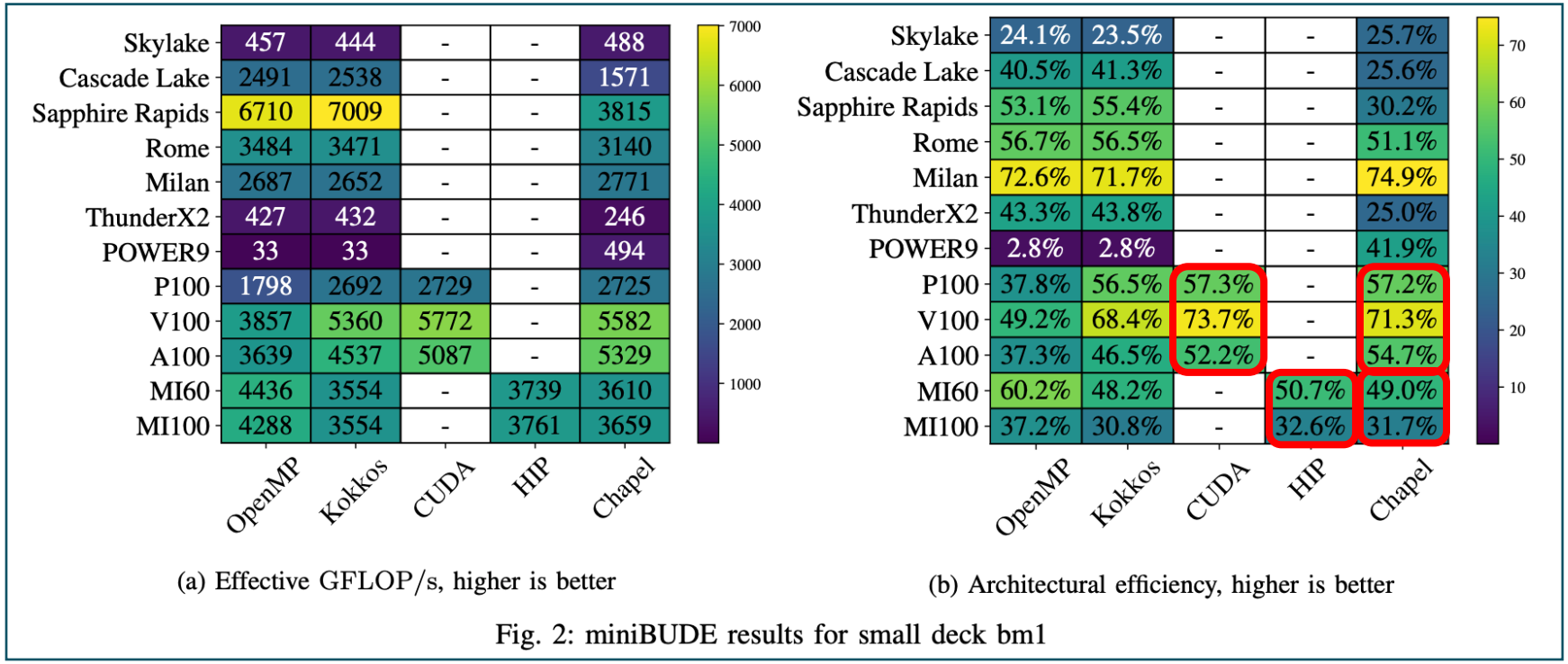
- We discussed Coral Reef application and showed its performance on Frontier
- In the follow slides we give performance results for a few additional miniapps/applications
  - Results were copied directly from the relevant papers (with the authors' permission)
- All these run on both NVIDIA and AMD GPUs and contain no vendor-specific code

- **MiniBude, BabelStream, and TeaLeaf**
  - Protein docking, STREAM benchmark, and heat conduction benchmark
  - Chapel implementations by Josh Milthorpe (Oak Ridge National Lab and Australian National University) et al.
  - Paper at 2024 Heterogeneity in Computing Workshop HCW (part of IPDPS)

- **ChOp** (Chapel Optimization)
  - Branch and bound optimization that does load balancing across nodes
  - Solves N-Queens but approach is generally applicable to optimization and search (e.g., operational scheduling)
  - Presented at EuroPar 2024 by Tiago Carneiro (Interuniversity Microelectronics Centre (IMEC), Belgium) et al.

MiniBude: Chapel implementation by Josh Milthorpe from ORNL

- MiniBude is miniapp of Bude (a protein docking simulation)
  - Main kernel is a triple nested loop over proteins, ligands, and poses, and computes energy for each ligand-protein pair
  - The computation is very arithmetically intensive and makes significant use of trigonometric functions
- For this miniapp, Chapel's performance is close to CUDA's and HIP's



(a) Effective GFLOP/s, higher is better

(b) Architectural efficiency, higher is better

Fig. 2: miniBUDE results for small deck bm1

Figure from: "Performance Portability of the Chapel Language on Heterogeneous Architectures". Josh Milthorpe (Oak Ridge National Laboratory, Australian National University), Xianghao Wang (Australian National University), Ahmad Azizi (Australian National University) Heterogeneity in Computing Workshop (**HCW**)

BabelStream: Chapel implementation by Josh Milthorpe from ORNL

- Performs stream triad computation computing A = B + α ∗C for arrays A, B, C and scalar α
- Chapel performs competitively for this benchmark

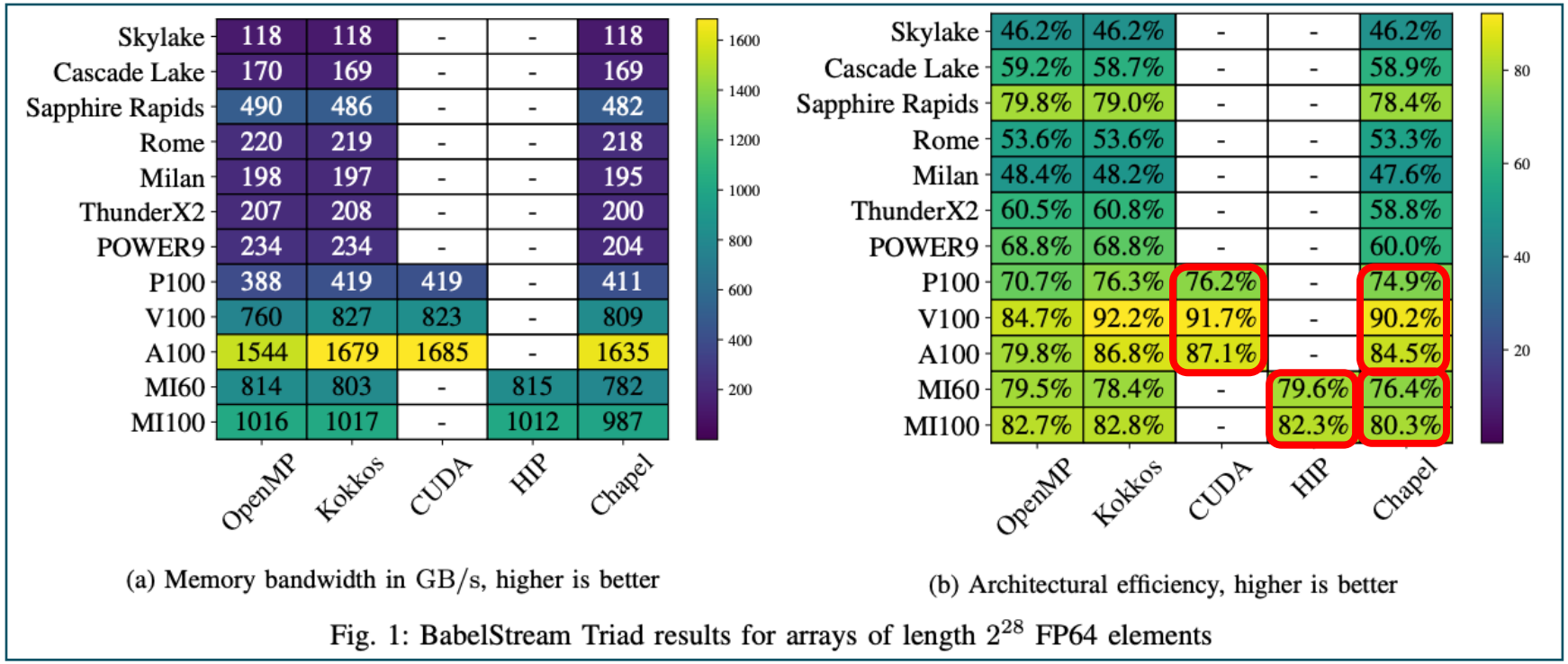


Figure from: "Performance Portability of the Chapel Language on Heterogeneous Architectures". Josh Milthorpe (Oak Ridge National Laboratory, Australian National University), Xianghao Wang (Australian National University), Ahmad Azizi (Australian National University) Heterogeneity in Computing Workshop (**HCW**)

ChOp: N-Queens Solver by Tiago Carneiro

- Results are shown for two different problem sizes for N-Queens, "21" and "22"
- The "CUDA"/"HIP" versions use Chapel's interoperability features to launch kernels written in CUDA/HIP
- For size=21 Chapel and CUDA/HIP perform similarly well, for size=22 the HIP version would crash so we don't have comparative results for that (the Chapel version would, however, scale)

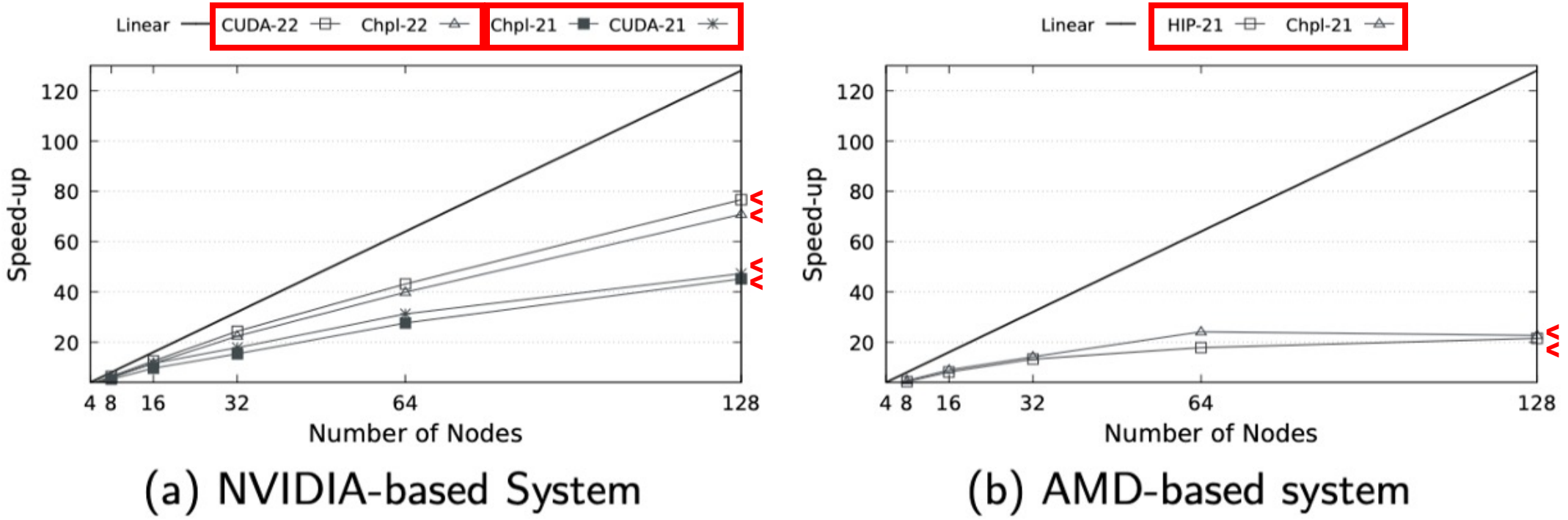


Figure from: "Investigating Portability in Chapel for Tree-Based Optimizations on GPU-powered Clusters". Tiago Carneiro, Engin Kayraklioglu, Guillaume Helbecque, Nouredine Melab

Time to Evolve Applications for Scientific Discovery

- Eric Laurendeau (PI) gave our CHIUW 2021 keynote
  - title: *HPC Lessons From 30 Years of Practice in CFD Towards Aircraft Design and Analysis*
  - quote:

    "So CHAMPS, that's the new solver that has been made, and all made by the students… So, [Chapel] promotes the programming efficiency. It was easy for them to learn. …I see the end result. We ask students at the master's degree to do stuff that would take 2 years and they do it in 3 months. And I'm not joking, this is from 2 years to 3 months. So if you want to take a summer internship and you say 'program a new turbulance model', well they manage. And before, it was impossible to do."

- Recent refactor of Arkouda commands API that reduces boilerplate code by about ~80x

Efforts Underway

- Time to try out a new technology
  - cloud images, map of how to install from source
  - Updating HPC carpentries entry for Chapel with author Alex Razoumov at Simon Frasier

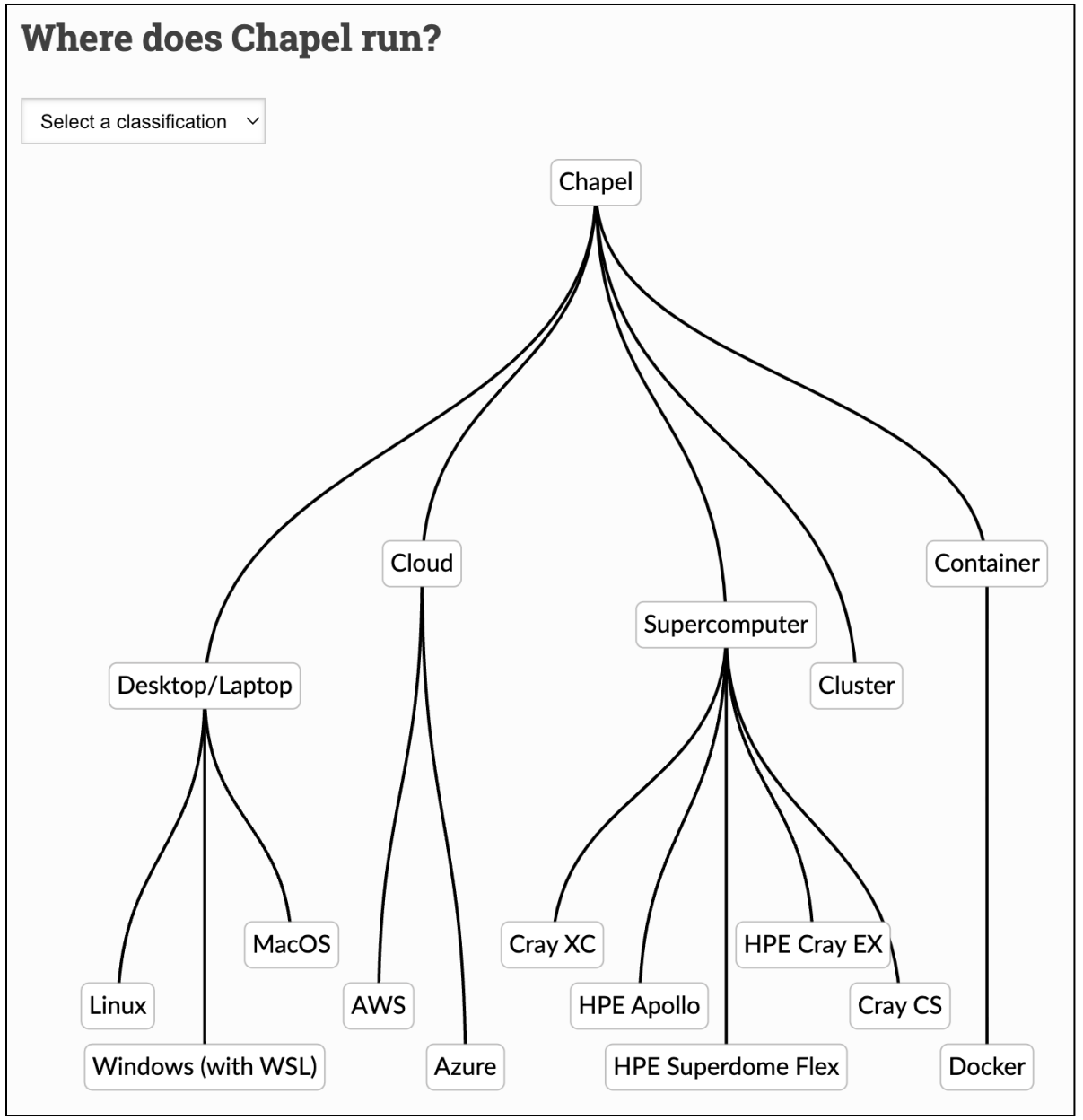
- Time to first usable application that solves their problem
  - Cloud images like AMIs for applications

- Ease of use by collaborators
  - Calling Chapel programs from Python and creating a PyPI package for installation from pip
  - Providing distributed arrays for Xarray Python package

Time to Tune Applications

- Installing from source enables configuration to best leverage network fabric and other architectural features

- Thus, mapping out places can install and best practices for configuration



Python Array API in Arkouda

Of interest to the PANGEO climate science community

- The Python *Array API* defines an interface for creating and manipulating multidimensional arrays
  - intended to be used and implemented by a variety of data-science / scientific / ML libraries
  - can be thought of as a simplified subset of NumPy's API

- XArray is an array library commonly used in the climate science community (affiliated with PANGEO)
  - similar functionality to NumPy and Pandas with support for named array dimensions
  - designed to operate as a wrapper around any array type that implements the Array API

- Implementing the Array API in Arkouda will:
  - allow Arkouda to interoperate with other data-science libraries that have Array API support, like XArray
  - provide some functionality needed to bring Arkouda's API into closer parity with NumPy's API

Tackling the GPU performance gap with Tealeaf

- On this application Chapel performed well on CPUs but not GPUs
- We're now studying this application to improve Chapel's GPU support
  - And suspect better in-kernel reduction support will help close the gap

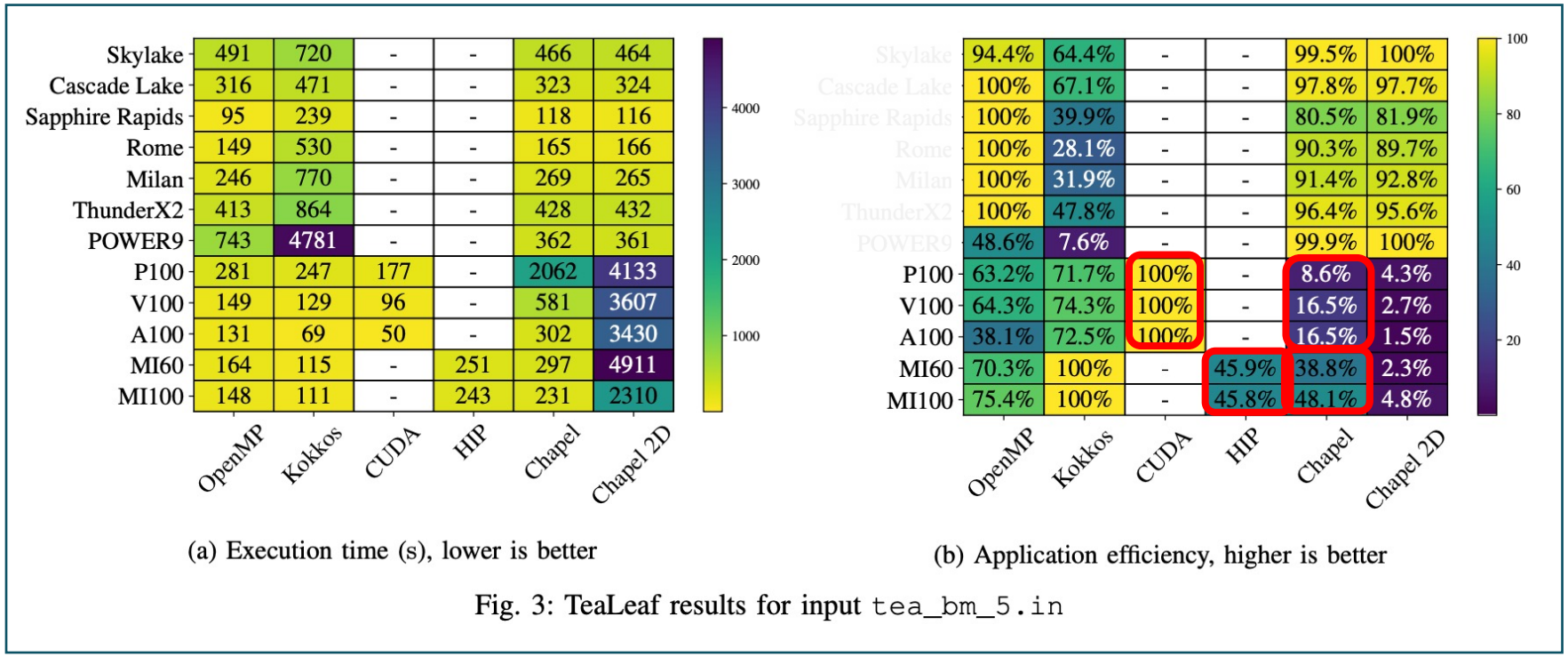


Figure from: "Performance Portability of the Chapel Language on Heterogeneous Architectures". Josh Milthorpe (Oak Ridge National Laboratory, Australian National University), Xianghao Wang (Australian National University), Ahmad Azizi (Australian National University) Heterogeneity in Computing Workshop (**HCW**)

Summary of Chapel Effectiveness for Scientific Application Development

- Real Applications, Real Fast with Chapel
  - Time to try out a new technology:
    –many rpms available, and GitHub spaces
  - Time to first usable application that solves their problem:
    –Examples requiring only ~3-6 months including learning to program in Chapel
  - Time to tune application for performance, scalability, and performance portability:
    –Computer Language Benchmarks Game shows multicore performance and code simplicity
    –Scaling over 8K nodes and 8TB/s for the radix sort in Arkouda
    –Performance portability over GPUs and CPUs shown with BabelSTREAM and MiniBude
  - Time to evolve the application for scientific discovery:
    –~3 months to add new physics to CHAMPS
  - Time to maintain application:
    –‘--warn-unstable’ compilation flag


Thank you
https://chapel-lang.org
@ChapelLanguage