Hewlett Packard
Enterprise

HPSF Project Proposal: The Chapel Programming Language

The Chapel Team

HPSF Technical Advisory Council meeting
January 9, 2025

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 and Productivity:

Chapel supports code that is as…

…**readable and writeable** as Python

…while also being as…

…**fast** as Fortran / C / C++

…**scalable** as MPI / SHMEM

…**portable** as C

…**GPU-ready** as CUDA / HIP / OpenMP / Kokkos / …

Matches HPSF's focus on:

- **lowering barriers to using HPC**
- **aiding HPC community growth**
- **enabling HPC development efforts**
- **portable software for diverse hardware**
- **performance and productivity**

Performance and Productivity: Arkouda Argsort

HPE Cray EX

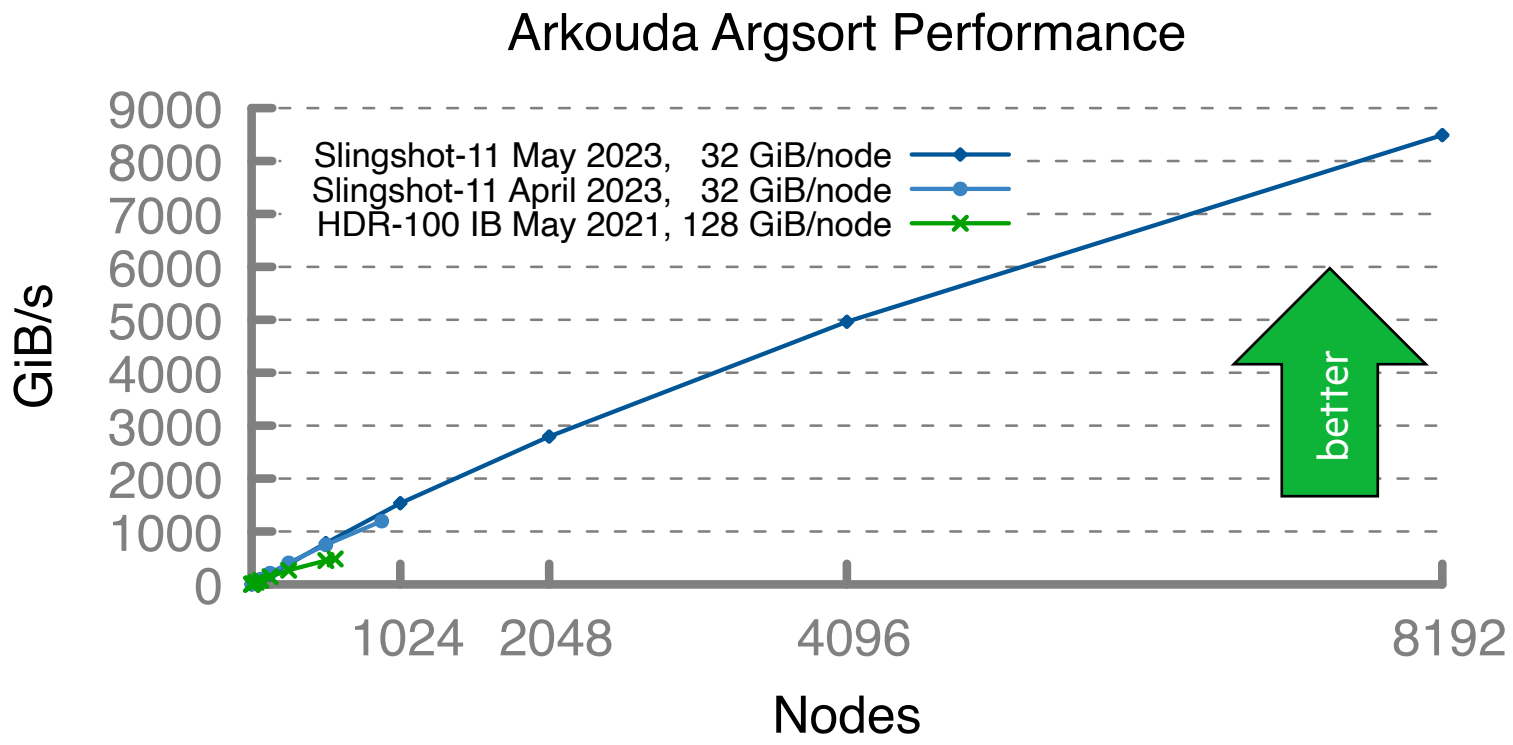
- Slingshot-11 network (200 Gb/s)
- **8192 compute nodes**
- **256 TiB** of 8-byte values
- ~8500 GiB/s **(~31 seconds)**

HPE Cray EX

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

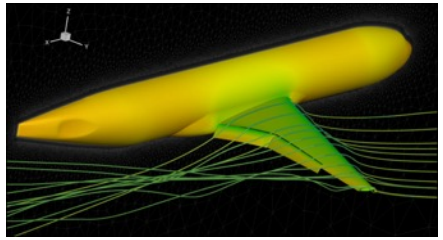
HPE Apollo

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

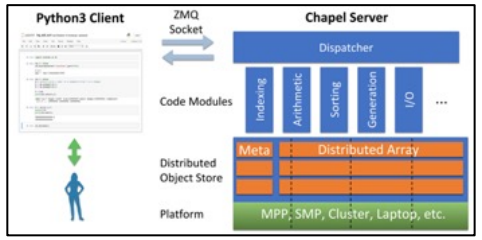


Implemented using ~100 lines of Chapel

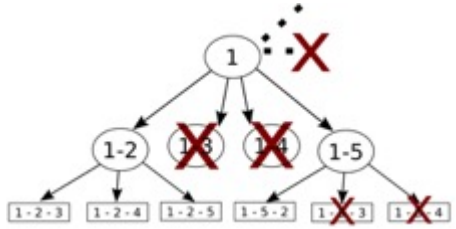
Applications of Chapel



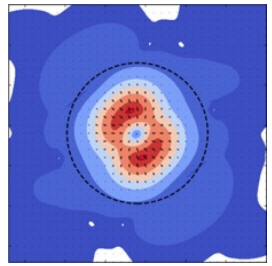
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



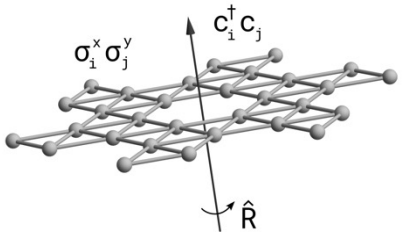
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



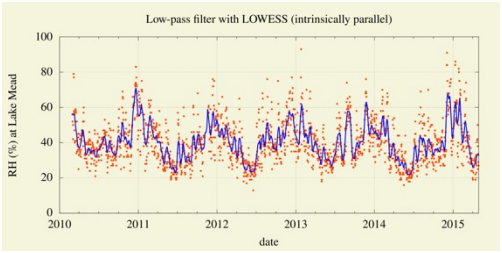
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

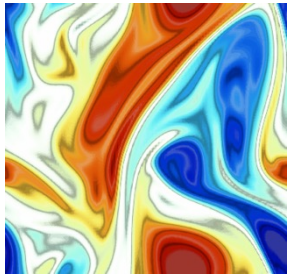


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

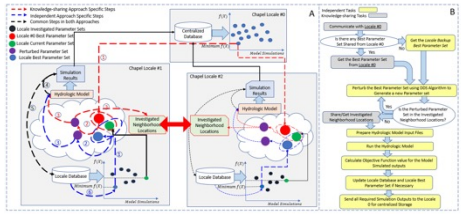


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

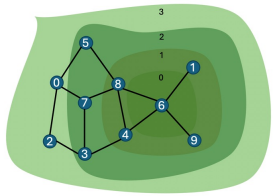
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



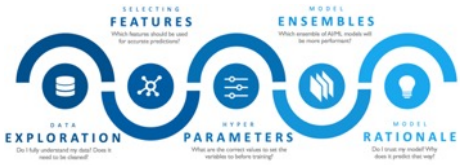
ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.



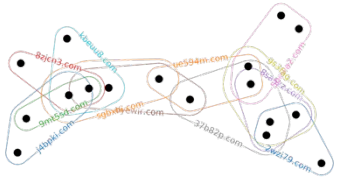
Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



Arachne Graph Analytics
Bader, Du, Rodriguez, et al.
New Jersey Institute of Technology



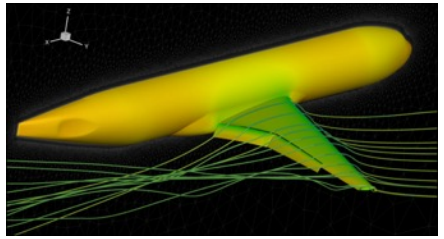
CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE



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

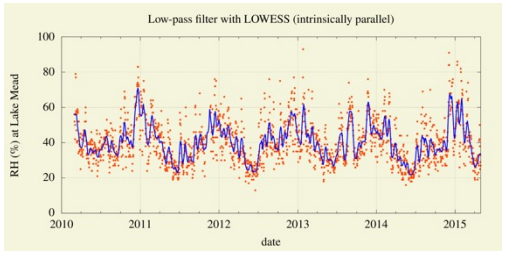
[images provided by their respective teams and used with permission]

Productivity Across Diverse Application Scales (code and system size)

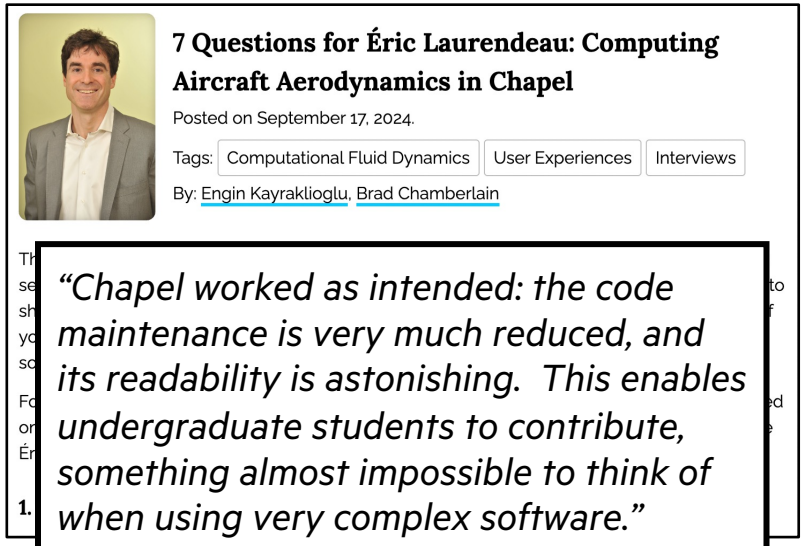


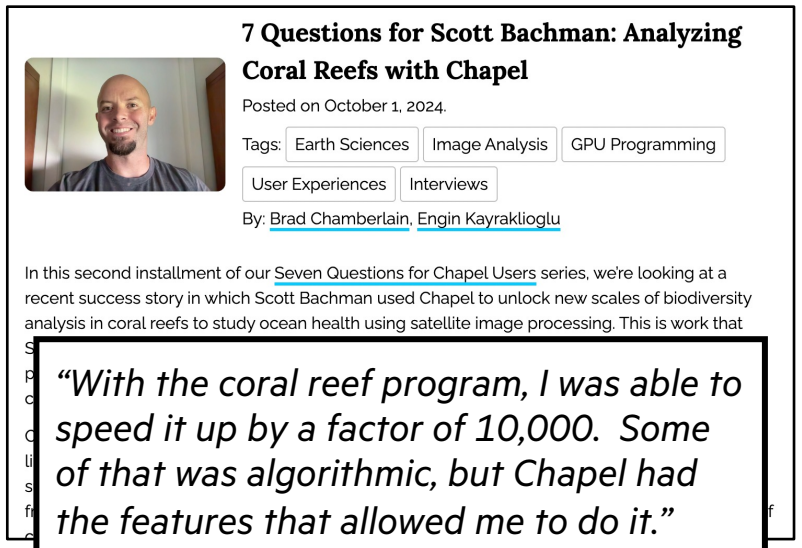
Computation: Aircraft simulation / CFD
Code size: 100,000+ lines
Systems: Desktops, HPC systems

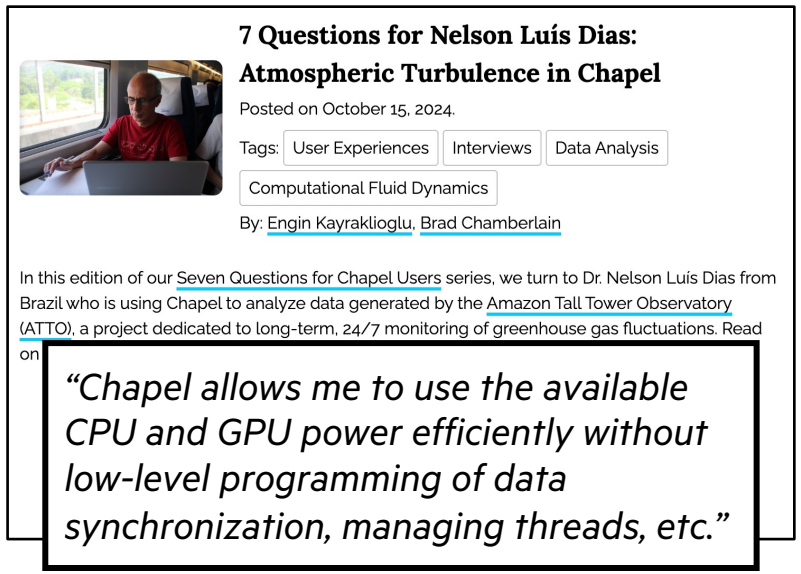
Computation: Coral reef image analysis
Code size: ~300 lines
Systems: Desktops, HPC systems w/ GPUs



Computation: Atmospheric data analysis
Code size: 5000+ lines
Systems: Desktops w/ GPUs







[read this interview series at: https://chapel-lang.org/blog/series/7-questions-for-chapel-users/]

Where Does Chapel Run?

In the Browser:

- GitHub Codespaces
- Attempt This Online (ATO)

Laptops/Desktops:

- Linux/UNIX
- Mac OS X
- Windows (leveraging WSL)

HPC Systems:

- Commodity clusters
- HPE/Cray supercomputers, such as:
  - Frontier
  - Perlmutter
  - Piz Daint
  - Polaris
  - …
- Other vendors' supercomputers

Cloud:

- AWS
- Microsoft Azure (?)
- Google Cloud (?)

CPUs:

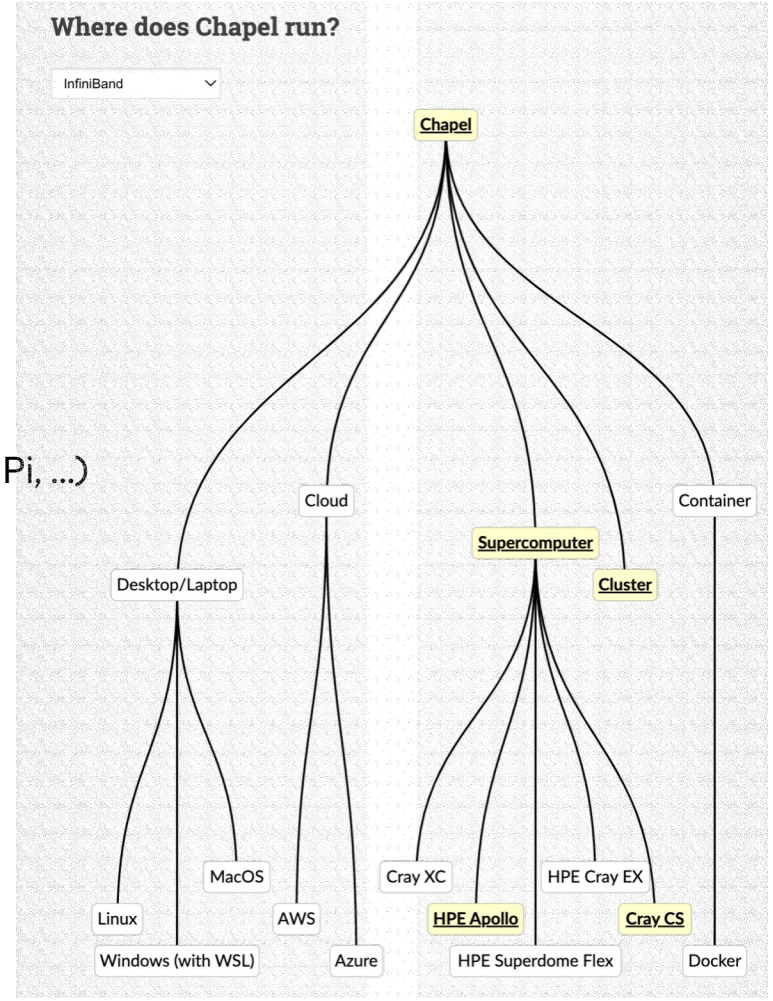
- Intel
- AMD
- Arm (M1/M2, Graviton, A64FX, Raspberry Pi, …)

GPUs:

- NVIDIA
- AMD

Networks:

- Slingshot
- Aries/Gemini
- InfiniBand
- AWS EFA
- Ethernet



https://chapel-lang.org/docs/usingchapel/portability.html

Synergies Between Chapel and HPSF's Goals

Lowering barriers to using HPC	• Chapel is helping users write real applications
	  • Many are writing HPC code for the first time
	  • others are simply leveraging their desktop multicore CPUs + GPUs
	  • Others are HPC experts, working more quickly than they otherwise could've

Aiding HPC community growth	
Enabling HPC development efforts	
Portable software for diverse hardware	• Chapel currently supports most any HPC, desktop, or cloud system • Its language design and code architecture support porting to others
Performance and productivity	• Chapel performance often matches or beats conventional HPC technologies • Code is almost always shorter and easier to read/write/maintain

How Is Chapel Developed?

Platform: GitHub

License: Apache 2.0

Release Cadence: Quarterly (Mar, June, Sept, Dec)

Contributors:

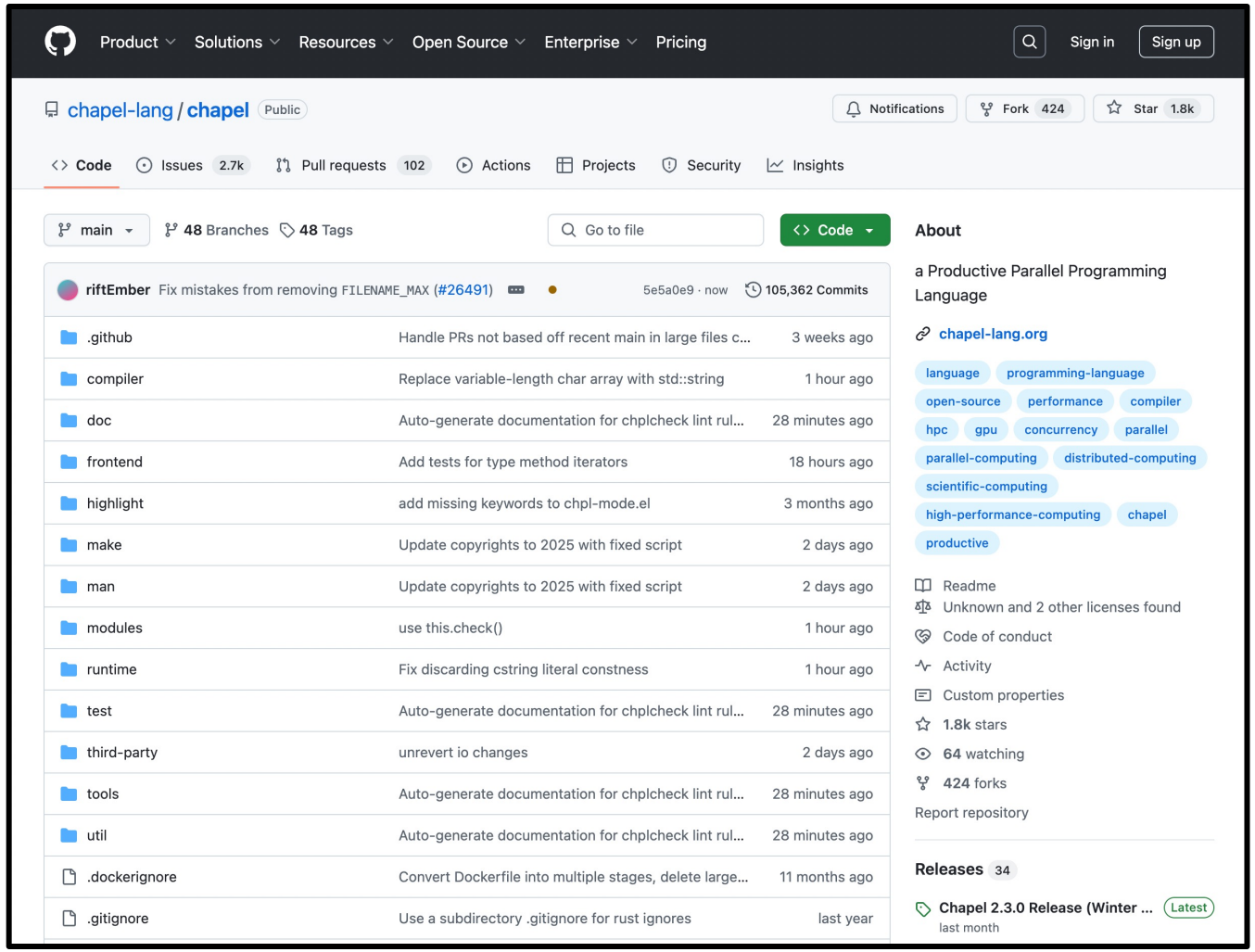
- over time: 200+ from 100+ affiliations worldwide
- per-release: ~25–35, primarily from HPE/Cray
- docs for contributors: chapel-lang.org/docs/developer/

Code of Conduct: CODE_OF_CONDUCT.md

Governance: HPE-led, with user input and guidance

Decision-Making:

- consensus-oriented
- ad hoc subteams to explore and propose solutions
- discussions on GitHub issues and in community forums



https://github.com/chapel-lang/chapel

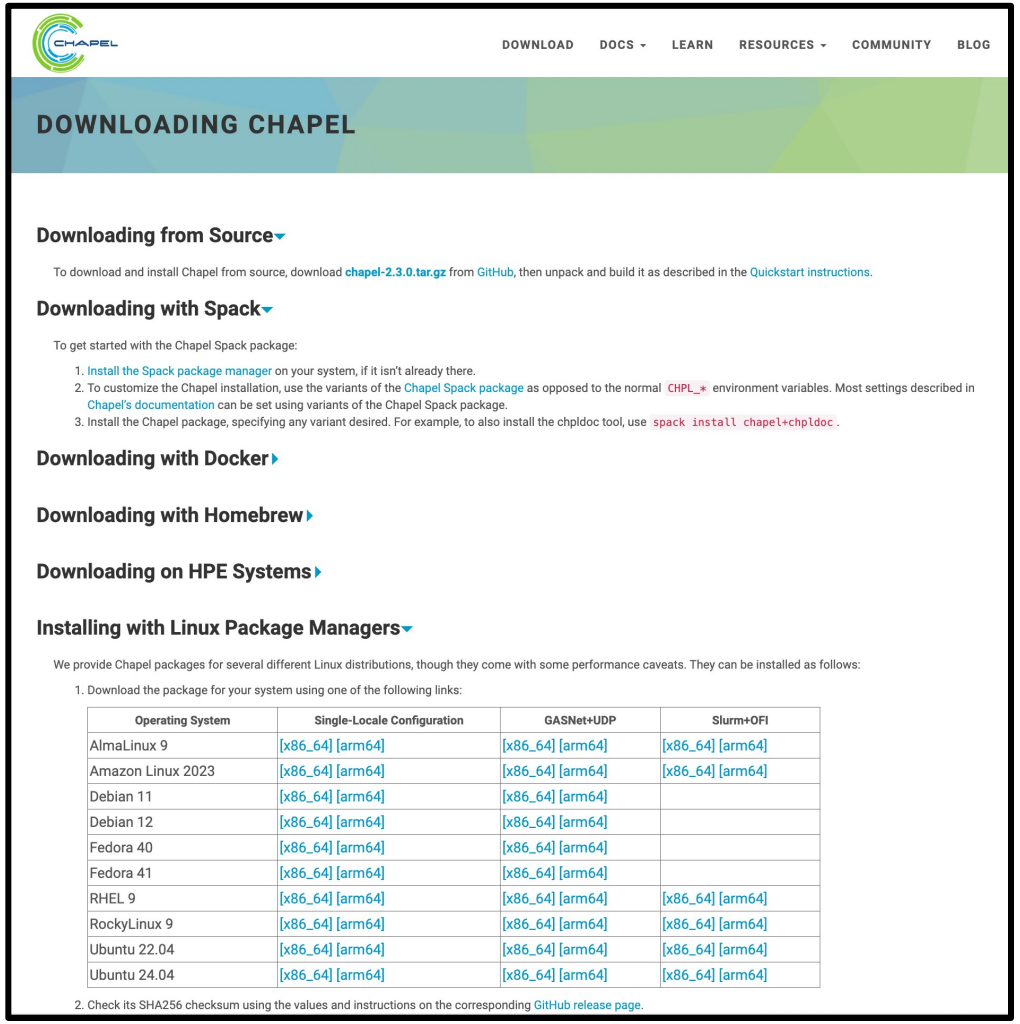
How Is Chapel Deployed?

Release Formats:

- Source releases via GitHub
- Spack
- E4S
- Linux packages via apt/rpm
- Homebrew
- Docker
- Modules on HPE Cray systems
- ATO / GitHub Codespaces

Chapel releases leverage and bundle:

- **GASNet** (LBNL) and **libfabric** (OFI) for communication
- **Qthreads** (Sandia) for tasking
- **hwloc** (OpenMPI) for HW introspection
- **jemalloc** for memory allocation
- **LLVM** for back-end compilation
- **GMP** for bigint support
- **re2** for regular expression support
- **libunwind**, **utf8-decoder**, **whereami**



https://chapel-lang.org/download/

How Is Chapel Tested?

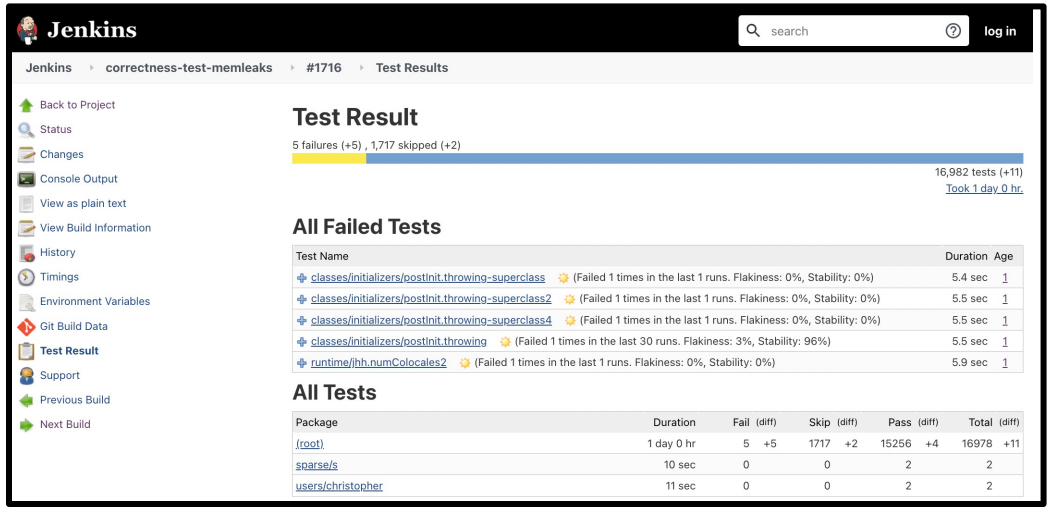
CI/CD: relatively quick, pre-merge checks

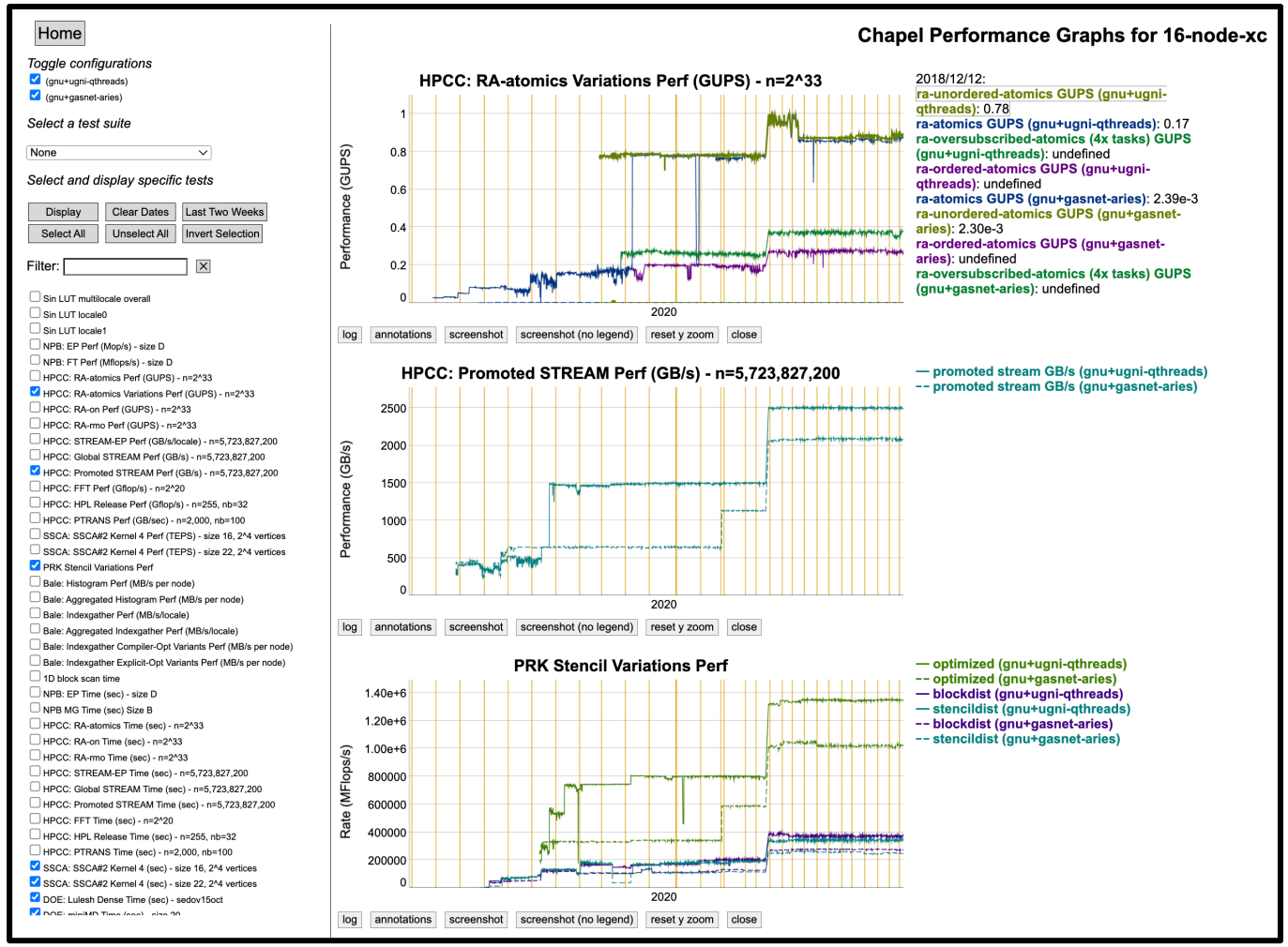
Smoke Testing: longer, post-merge checks

- goal: head off catastrophes overnight

Nightly Testing: 125+ jobs managed with Jenkins

- leverages a suite of 17,500+ tests
- spans a multitude of platforms, vendors, networks, …
- jobs for specific configs, compiler flags, user apps, …
- jobs for memory leaks, valgrind/asan errors, …
- jobs for correctness and performance tracking





https://chapel-lang.org/perf-nightly.html

How Does the Chapel Community Communicate?

Live/Virtual Events

- ChapelCon (formerly CHIUW), annually
- Office Hours, monthly
- Live Demo Sessions, monthly

Electronic Broadcasts

- Chapel Blog, ~biweekly
- Community Newsletter, quarterly
- Announcement Emails, around big events

Community / User Forums

- Discord
- Discourse
- Email Contact Alias `chapel+qs@discoursemail.com`
- GitHub Issues
- Gitter
- Reddit
- Stack Overflow

Social Media

- Bluesky
- Facebook
- LinkedIn
- Mastodon
- X / Twitter
- YouTube

Chapel's HPSF Application

Maturity Level:

- We're applying at the "Established" stage, believing we easily meet the requirements
- We're interested in improving our processes under HPSF, which would also help move us toward the "Core" stage
  - establishing a community governing body and documenting it
  - establishing security processes
  - extending merge privileges to non-HPE developers

Motivations for Applying:

- Believe it will help improve our project's visibility and stature
- Expect it to help address "single-vendor" concerns that discourage potential users and collaborators
- Hope to network with other open-source HPC projects and share best practices
  - Our experience with HPC testing, portability, performance tracking, etc. may be useful to other projects, and we have lots to learn too
  - We're particularly interested in leveraging Linux Foundation / HPSF expertise in the "Core" areas noted above

Infrastructure Needs:

- Nothing pressing at present
- Ideally: Compute resources outside HPE for community testing or the ability to "try Chapel in the cloud"
- Might be nice: Assistance with things like paid CI/CD runners, DNS registration, financial donations, etc.

Thank you

https://chapel-lang.org
@ChapelLanguage