



**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

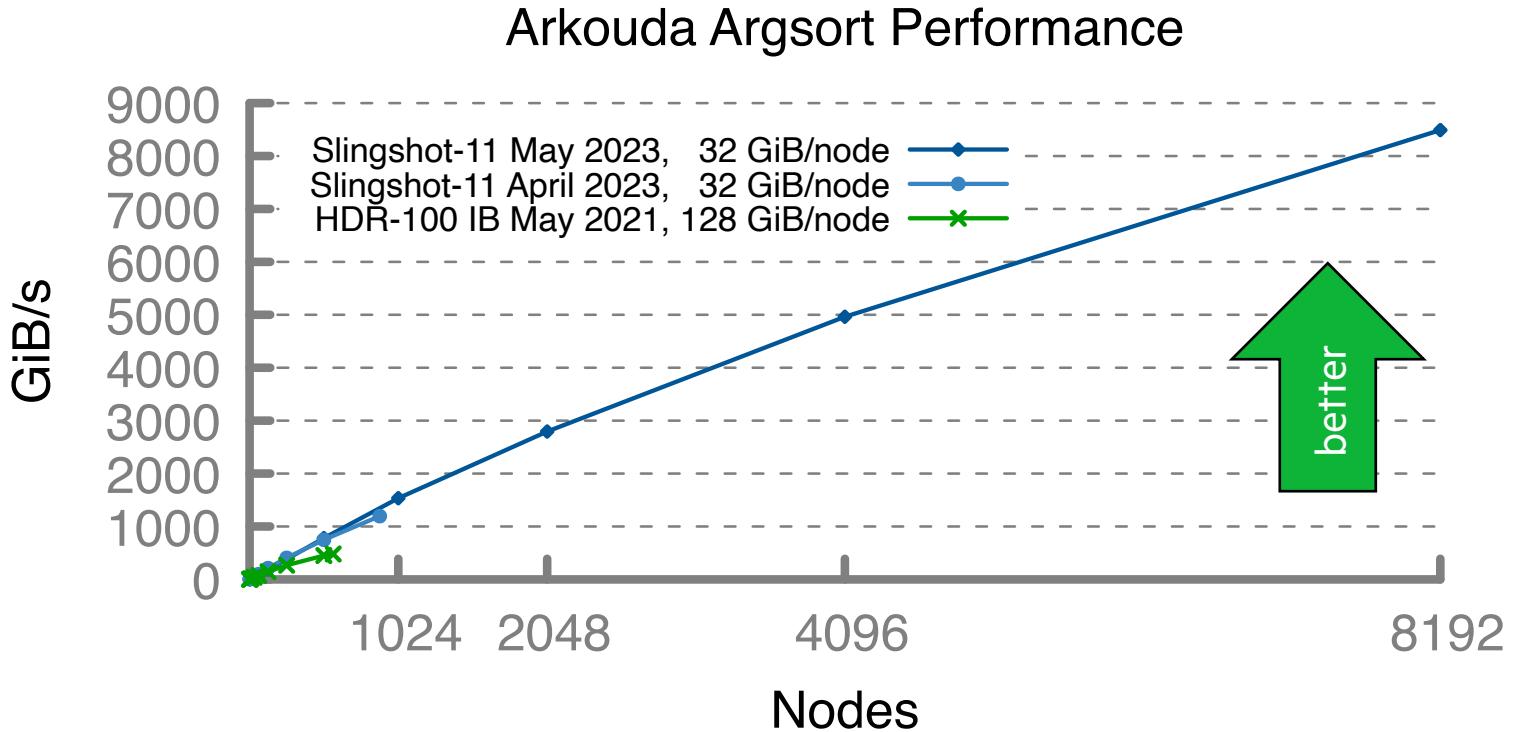
- Slingshot-11 network (200 Gb/s)
- **8192 compute nodes**
- **256 TiB** of 8-byte values
- ~8500 GiB/s (**~31 seconds**)

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

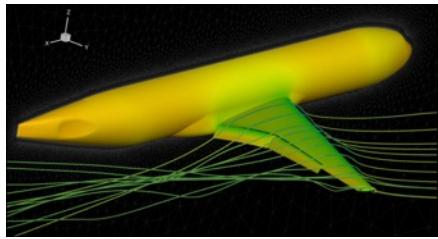
- HDR-100 InfiniBand network (100 Gb/s)
- 576 compute nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)



Implemented using ~100 lines of Chapel



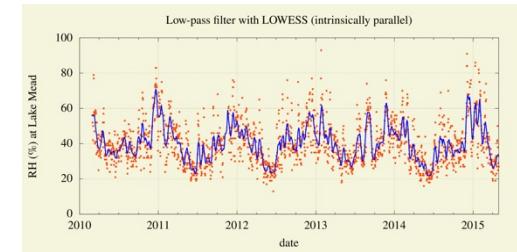
Productivity Across Diverse Application Scales (code and system size)



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



7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel

Posted on September 17, 2024.

Tags: Computational Fluid Dynamics, User Experiences, Interviews
By: [Engin Kayraklıoglu](#), [Brad Chamberlain](#)

"Chapel worked as intended: the code maintenance is very much reduced, and its readability is astonishing. This enables undergraduate students to contribute, something almost impossible to think of when using very complex software."



7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel

Posted on October 1, 2024.

Tags: Earth Sciences, Image Analysis, GPU Programming, User Experiences, Interviews
By: [Brad Chamberlain](#), [Engin Kayraklıoglu](#)

In this second installment of our [Seven Questions for Chapel Users](#) series, we're looking at a recent success story in which Scott Bachman used Chapel to unlock new scales of biodiversity analysis in coral reefs to study ocean health using satellite image processing. This is work that

"With the coral reef program, I was able to speed it up by a factor of 10,000. Some of that was algorithmic, but Chapel had the features that allowed me to do it."



7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel

Posted on October 15, 2024.

Tags: User Experiences, Interviews, Data Analysis, Computational Fluid Dynamics
By: [Engin Kayraklıoglu](#), [Brad Chamberlain](#)

In this edition of our [Seven Questions for Chapel Users](#) series, we turn to Dr. Nelson Luis Dias from Brazil who is using Chapel to analyze data generated by the [Amazon Tall Tower Observatory \(ATTO\)](#), a project dedicated to long-term, 24/7 monitoring of greenhouse gas fluctuations. Read on

"Chapel allows me to use the available CPU and GPU power efficiently without low-level programming of data synchronization, managing threads, etc."

[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:

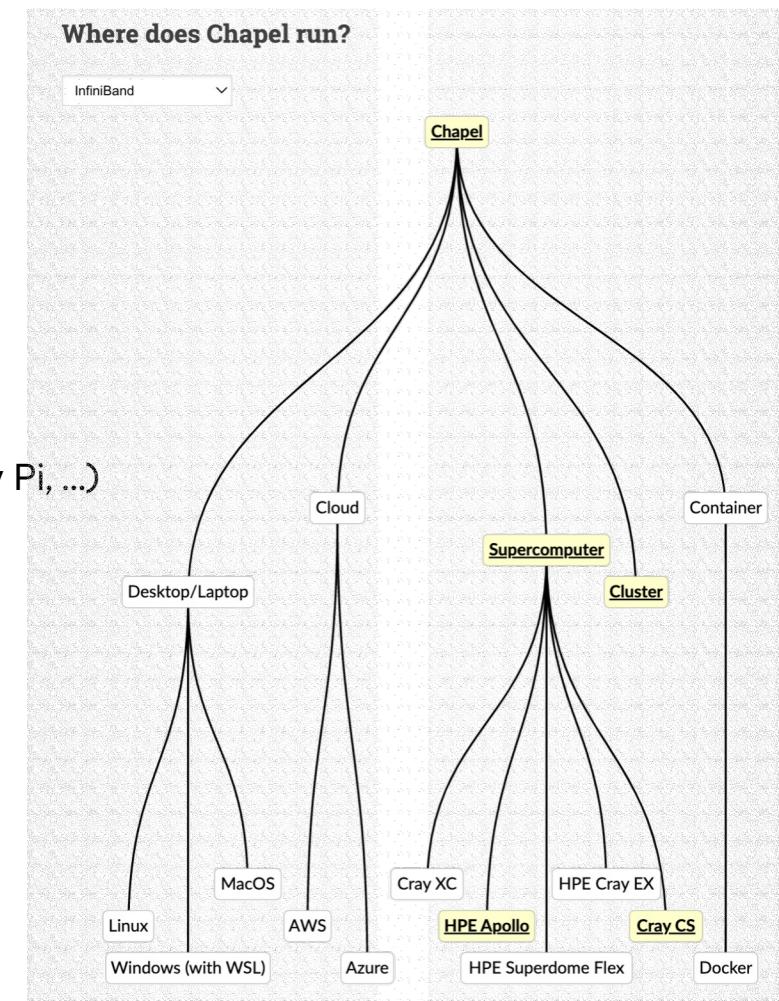
- Intel
- AMD
- Arm (M1/M2, Graviton, A64FX, Raspberry Pi, ...)

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	<ul style="list-style-type: none">• Chapel is helping users write real applications<ul style="list-style-type: none">• Many are writing HPC code for the first time<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Chapel currently supports most any HPC, desktop, or cloud system• Its language design and code architecture support porting to others
Performance and productivity	<ul style="list-style-type: none">• 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

The screenshot shows the GitHub repository page for `chapel-lang / chapel`. The repository is public and has 105,362 commits across 48 branches and 48 tags. The main branch is current. Recent commits include fixes for FILENAME_MAX (#26491), updates to .github, compiler, and documentation, and auto-generate documentation for chplcheck lint rules. The repository has 1.8k stars, 64 watchers, and 424 forks. It is described as a Productive Parallel Programming Language and includes tags for language, programming-language, open-source, performance, compiler, hpc, gpu, concurrency, parallel, parallel-computing, distributed-computing, scientific-computing, high-performance-computing, chapel, and productive. A link to `chapel-lang.org` is provided. The releases section shows Chapel 2.3.0 Release (Winter ...) from last month.

<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**

The screenshot shows the "DOWNLOADING CHAPEL" section of the Chapel website. It includes links for "Downloading from Source", "Downloading with Spack", "Downloading with Docker", "Downloading with Homebrew", and "Downloading on HPE Systems". Below these, there is a table for "Installing with Linux Package Managers" and instructions for checking SHA256 checksums.

Operating System	Single-Locale Configuration	GASNet+UDP	Slurm+OFI
AlmaLinux 9	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]
Amazon Linux 2023	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]
Debian 11	[x86_64] [arm64]	[x86_64] [arm64]	
Debian 12	[x86_64] [arm64]	[x86_64] [arm64]	
Fedor a 40	[x86_64] [arm64]	[x86_64] [arm64]	
Fedor a 41	[x86_64] [arm64]	[x86_64] [arm64]	
RHEL 9	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]
RockyLinux 9	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]
Ubuntu 22.04	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]
Ubuntu 24.04	[x86_64] [arm64]	[x86_64] [arm64]	[x86_64] [arm64]

<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

 Jenkins

Jenkins > correctness-test-memleaks > #1716 > Test Results

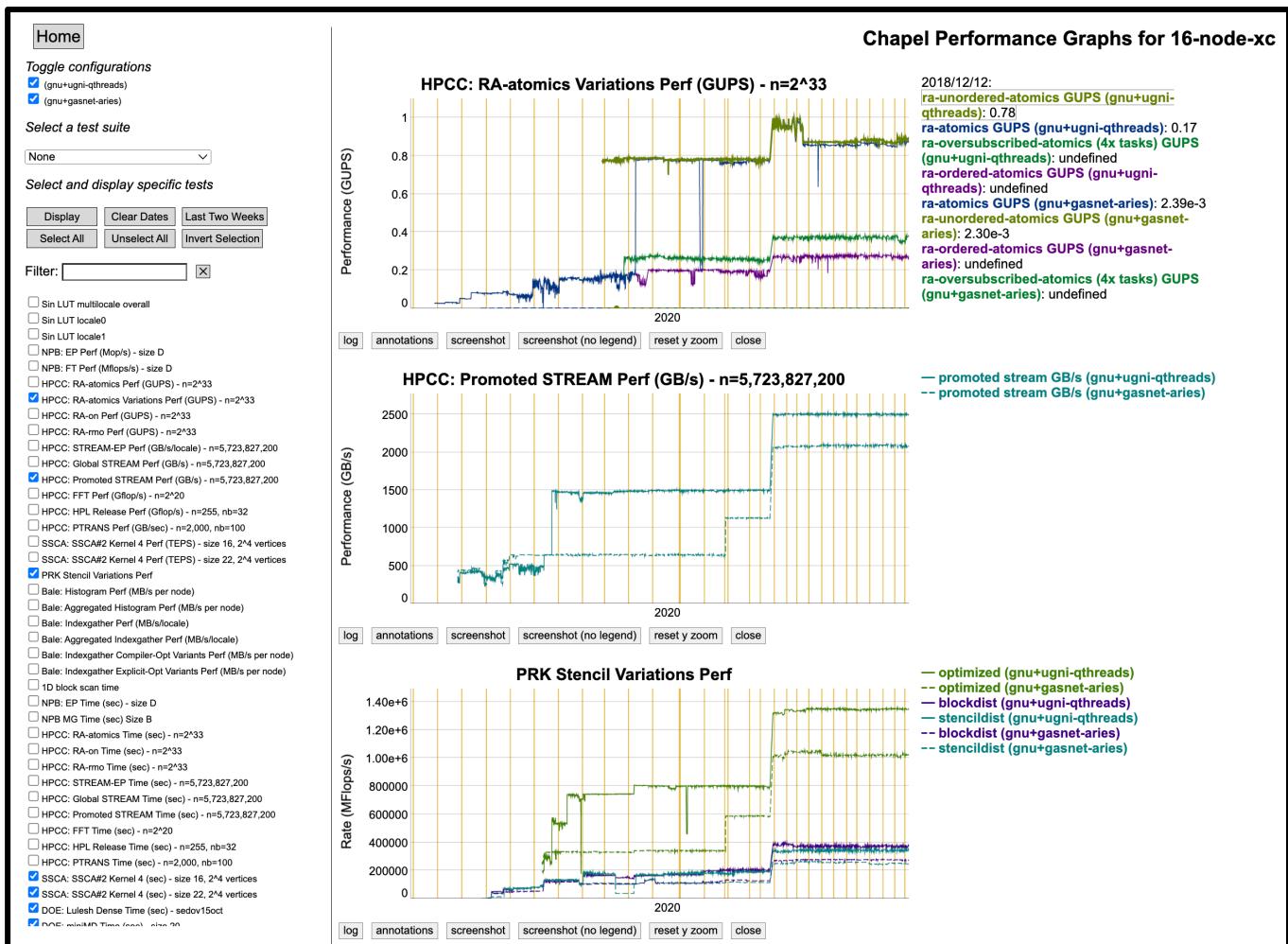
Test Result
5 failures (+5), 1,717 skipped (+2)
16,982 tests (+11) Took 1 day 0 hr.

All Failed Tests

Test Name	Duration	Age
classes/initializers/postinit.throwing-superclass	5.4 sec	1
classes/initializers/postinit.throwing-superclass2	5.5 sec	1
classes/initializers/postinit.throwing-superclass4	5.5 sec	1
classes/initializers/postinit.throwing	5.5 sec	1
runtim/libh.numColocales2	5.9 sec	1

All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
(root)	1 day 0 hr	5 +5	1717 +2	15256 +4	16978 +11
sparse/s	10 sec	0	0	2	2
users/christopher	11 sec	0	0	2	2



<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](#)
chapel+qs@discoursemail.com
- Email Contact Alias
- [GitHub Issues](#)
- [Gitter](#)
- [Reddit](#)
- [Stack Overflow](#)



chapel+qs@discoursemail.com



GITTER



stackoverflow

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

