HPE

# ChapelCon '25: State of the Chapel Project

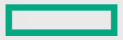
Brad Chamberlain

October 10, 2025

# Outline

- Summary of Technical Progress since ChapelCon ‘24
- Community Updates and News
- Chapel and HPSF
- Closing Thoughts

# Chapel 2.0
# (1 ½ years later)



# Chapel 2.0 (1½ Years Later)

— March 2024's Chapel 2.0 release was a milestone releasing, stabilizing core language and library features

— Since then, releases have continued on our quarterly cadence:

- **Chapel 2.1**: June 2024
- **Chapel 2.2**: Sept 2024
- **Chapel 2.3**: Dec 2024
- **Chapel 2.4**: Mar 2024
- **Chapel 2.5**: June 2025
- **Chapel 2.6**: Sept 2025

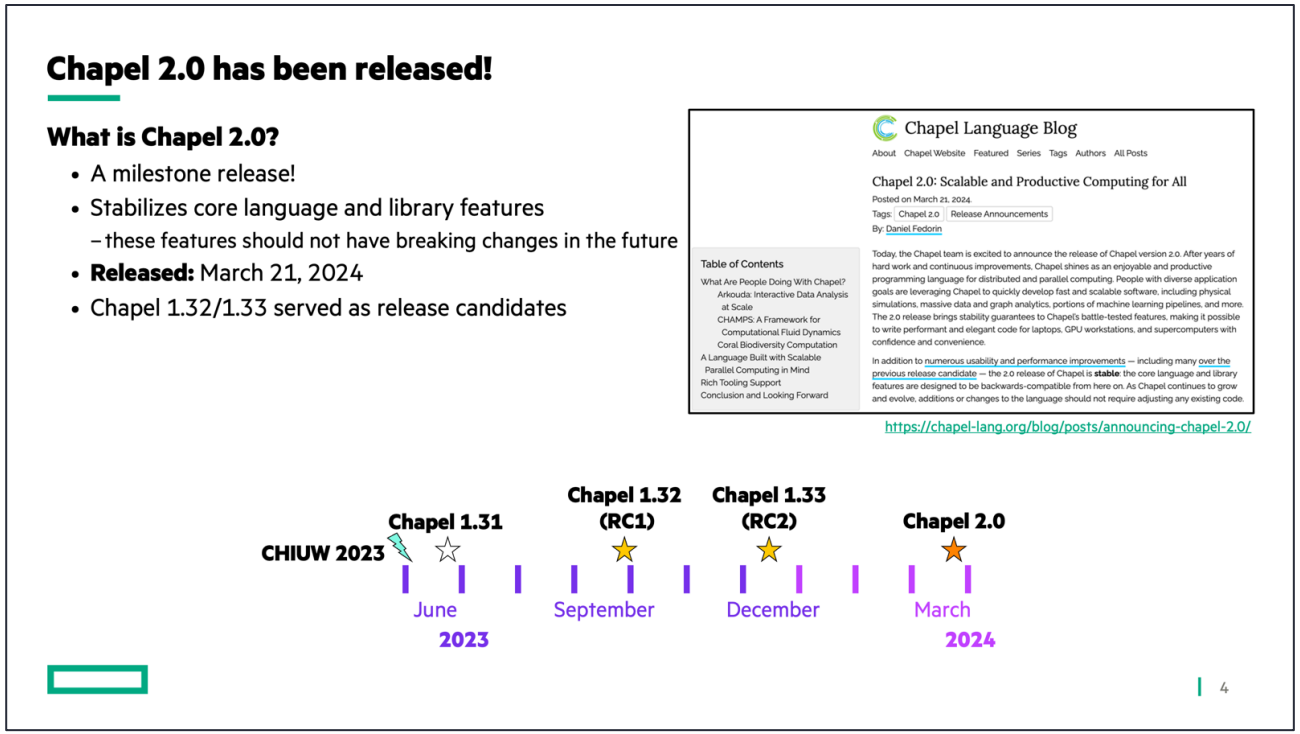
— And happily, stability has been maintained!

- many bugs have been fixed, and new features added
- we've also added a way to try breaking changes:

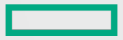
— Chapel editions:

- a way to opt into new features that alter behavior
- e.g. '—edition=preview' enables:
  - updating array 'reshape()' to support aliasing
  - improving the printing of 'complex' w/ NaNs
  - removing domain '.sorted()' iterators

Slide from ChapelCon '24:



# Language / Library Highlights Since 2.0



# New post-2.0 Features

## Language:

- **Remote variable declarations**

```
on remoteLocale var x: int;   // allocates 'x' on the remote locale but without introducing a new lexical scope
```

- **Multidimensional array literals** (designed with significant community input and involvement)

```
[1, 2, 3;
 4, 5, 6]                     // this is a 2x3 array over the indices {0..1, 0..2}
```

- **GPUs:** performance, flexibility, quality-of-life, and portability improvements
- **Improved sparse capabilities:** new queries, capabilities, optimizations (but more work remains)

**14:20 - 14:40** **Towards A General Aggregation Framework in Chapel**
*Oliver Alvarado Rodriguez, Engin Kayraklioglu, Bartosz Bryg, Mohammad Dindoost, David Bader and Brad Chamberlain*
▶ Description

## Library Modules:

- **Sort:**
  - stabilized the module and promoted it to a standard module
  - added a fast new scalable distributed 'sort()' routine
- **Python:** for calling from Chapel to Python
- **DynamicLoading:** for calling from Chapel to dynamic C/C-like libraries
- **Image:** for reading / writing image files from Chapel

**12:00 - 12:20** **Distributed-Memory Sorting in the Chapel Standard Library**
*Michael Ferguson*
▶ Description

**12:20 - 12:40** **Comparing Distributed-Memory Programming Frameworks with Radix Sort**
*Shreyas Khandekar and Matt Drozt*
▶ Description

**11:15 - 11:35** **If it walks like Python and quacks like Python, it must be….Chapel?**
*Jade Abraham and Lydia Duncan*
▶ Description

# Tools

# Coding Tools

- **VSCode:** task providers for compiling, running, debugging Chapel programs
- **chplcheck** (linter)**:** added and doc'd rules; improved ability to add new ones
- **chpldoc** (code-based documentation generator)**:**
- **chpl-language-server** (editor intelligence)**:**
- **chapel-py** (Python bindings to compiler front-end)**:**
- **mason** (package manager)**:**

continual improvements based on use and experience

- **Dyno** (front-end compiler rework and library): can now resolve and lower much more of the language
  - and often more correctly…
  - powers most of the tools above

# Debugging Tools

— **in general:** improved debug codegen
— **address sanitizers:** improved support for Chapel programs
— **documentation:** captured best practices

**Also...**

# Debugging: New LLDB pretty-printers for Chapel types

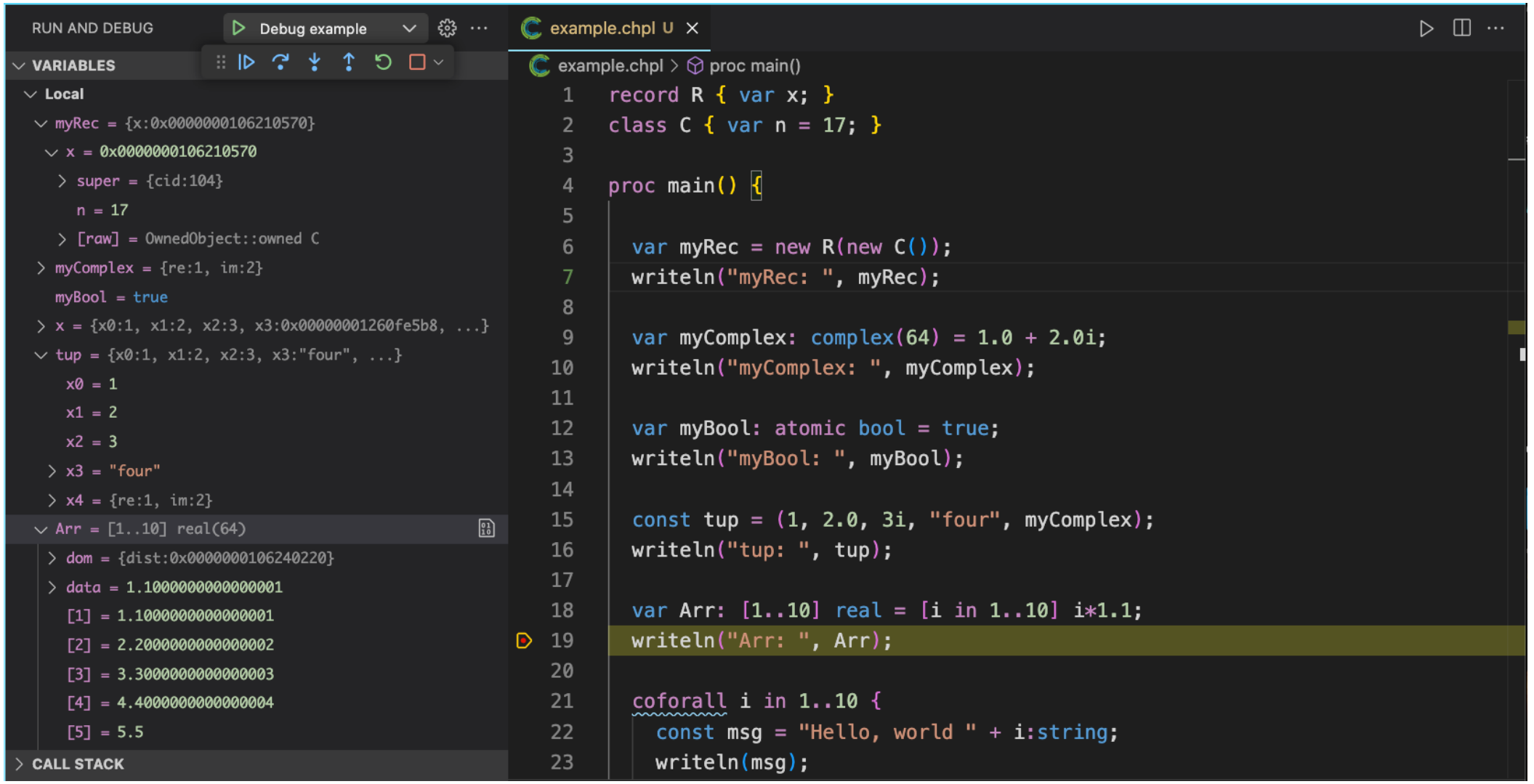
## Chapel 2.5:

```
(lldb) p myStr
(string) {
  buffLen = 13
  buffSize = 14
  cachedNumCodepoints = 13
  buff = 0x0000000106fd75d6 "Hello, world!"
  isOwned = false
  hasEscapes = false
  locale_id = 0
}
(lldb) p myDom
(_domain_DefaultRectangularDom_2_int64_t_positive) {
  _pid = -1
  _instance = 0x0000000106de0600
  _unowned = false
}
(lldb) p myArr2d
(_array_DefaultRectangularArr_2_int64_t_positive_int64_t_int64_t) {
  _pid = -1
  _instance = 0x000000010a7480a0
  _unowned = false
}
```

## Chapel 2.6:

```
(lldb) p myStr
(string) "Hello, world!" {
  size = 13
}
(lldb) p myDom
(_domain_DefaultRectangularDom_2_int64_t_positive) {1..10, 1..10} {
  dim = 1..10, 1..10 {}
}
(lldb) p myArr2d
(_array_DefaultRectangularArr_2_int64_t_positive_int64_t_int64_t) [1..10, 1..10]
 int64_t {
  dom = 0x00000001061fc600
  data = 0x0000000106568000
  [1,1] = 1
  [1,2] = 2
  [1,3] = 3
  [1,4] = 4
  [1,5] = 5
  [1,6] = 6
  [1,7] = 7
  [1,8] = 8
  [1,9] = 9
  [1,10] = 10
  [2,1] = 11
  [2,2] = 12
  [2,3] = 13
  [2,4] = 14
```

# Debugging: Integrated VSCode Support

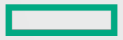


# Debugging: Prototype ‘chpl-parallel-dbg’ for multi-locale

```
(lldb) on 0
(lldb) f
frame #1: 0x0000000000a35e10 example_real`on_fn_chpl199(arr=0x00007f5d4e1fcf48, [illegible]
7f5d4e1fcf20, _coforallCount=__wide__EndCount_AtomicT_int64_t_int64_t @ 0x00007f5d4e1fce
b8) at example.chpl:10
   7        writeln("Hello from locale ", myVar);
   8        const mySlice = arr[arr.localSubdomain()];
   9        writeln("My slice is: ", mySlice);
-> 10       import Debugger; Debugger.breakpoint;
   11     }
   12   }
(lldb) p myVar
(long) 0
(lldb) p mySlice
(ChapelArray::[domain(1,int(64),one)] int(64)) {
  _pid = -1
  _instance = {
    locale = {}
    addr = 0x00007f5d4eb58000
  }
  _unowned = false
}
(lldb) c
Process 609716 resuming
Target 1: (example_real) stopped.
(lldb) on 1
(lldb) p myVar
(long) 1
(lldb) p mySlice
(ChapelArray::[domain(1,int(64),one)] int(64)) {
  _pid = -1
  _instance = {
    locale = {}
    addr = 0x00007f07e8359180
  }
  _unowned = false
}
(lldb)
```

custom ‘on’ command supports switching between locales by ID

# Packaging / Portability



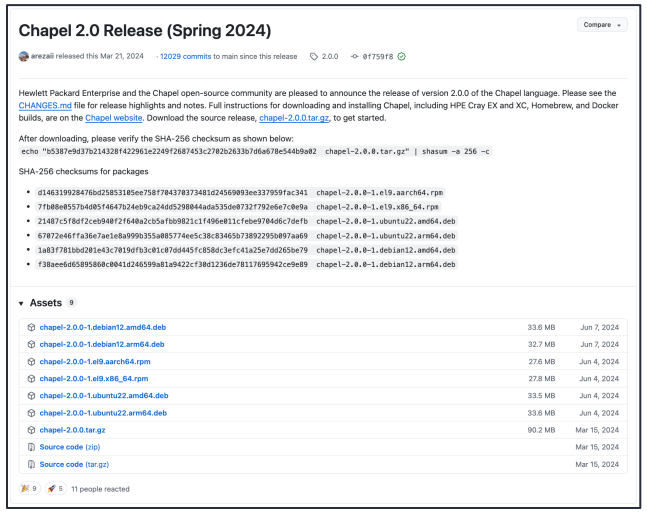
# Packaging / Portability Improvements

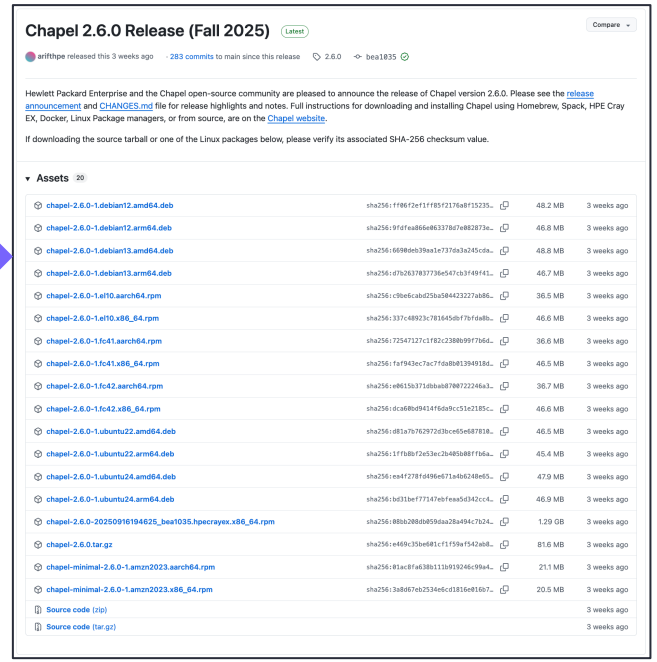
- A new Chapel Spack package
  - Also a pair of new packages for the Arkouda server and client
- Several new Linux package releases and configs:
  - AlmaLinux 10
  - Amazon Linux 2023
  - Debian 12
  - Debian 13
  - Fedora 41
  - Fedora 42
  - RHEL 10
  - RockyLinux 10
  - Ubuntu 22.04
  - Ubuntu 24.04
- Many improvements to the Homebrew release
- Improved AWS / EFA support
- Improved access to the HPE Cray EX RPM via GitHub and My HPE Software Center

**Spack: The Community's Road to the HPSF and Version 1.0**

**Todd Gamblin, LLNL & HPSF, Invited Talk**

The past year has been transformative for the twelve-year-old Spack project, starting with its inclusion in the High Performance Software Foundation (HPSF) and culminating in its 1.0 release. Spack v1.0, released in July, is the first version to offer a stable package API and to integrate true compiler dependencies into its core model—features developed over many years. This talk will cover how the Spack community evolved to this point and detail the decision-making process behind joining the HPSF and finally taking the plunge and going 1.0.



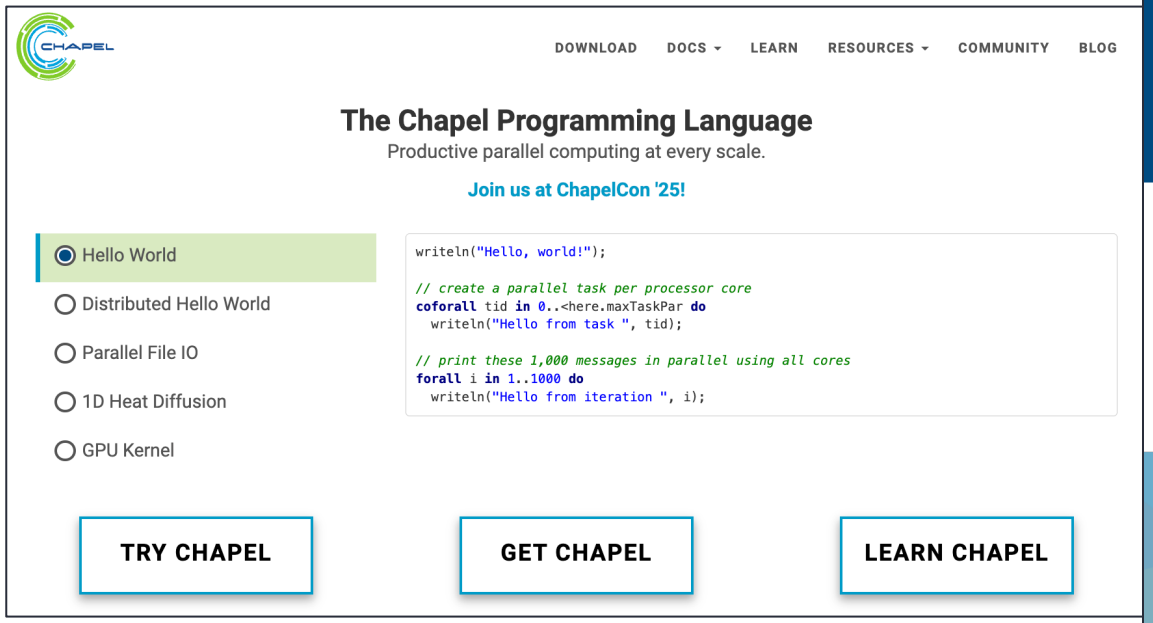



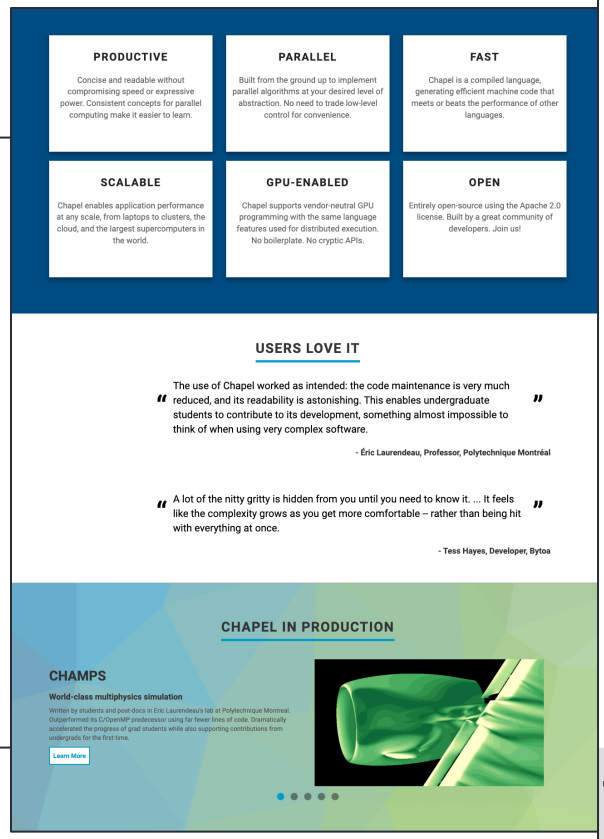
# Community Highlights

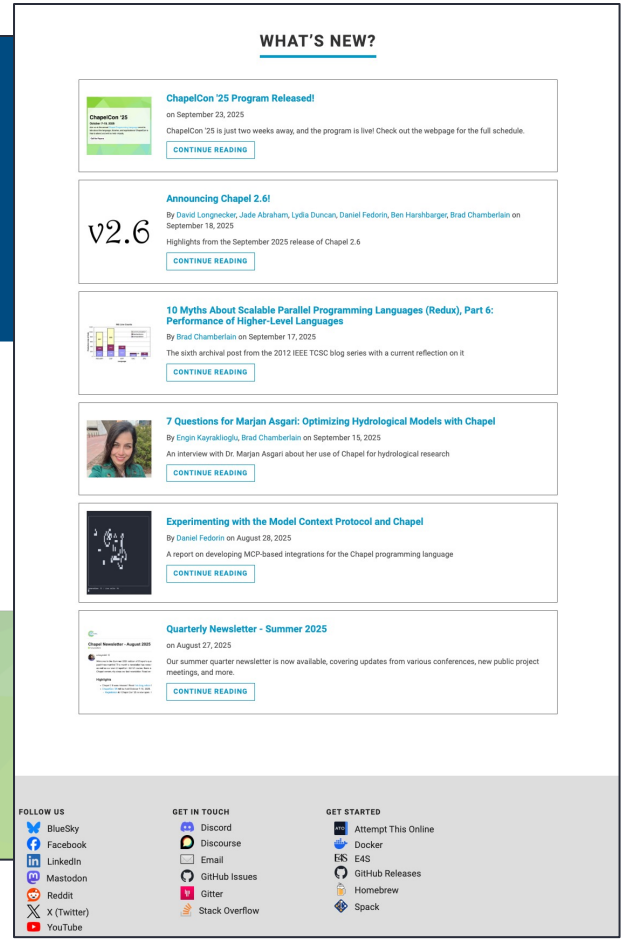
# A New Website!

We launched our new website on Jan 8!

- And the website's repo is now public!



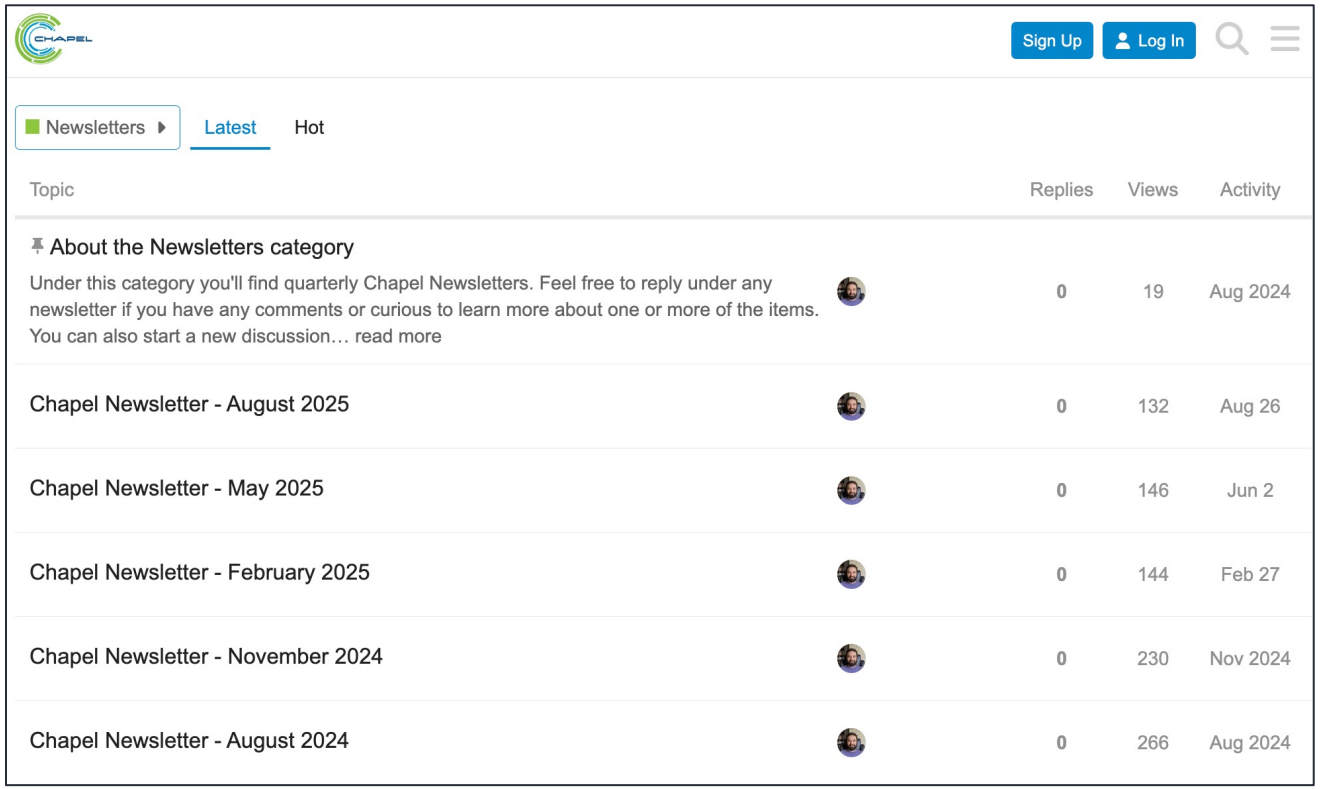



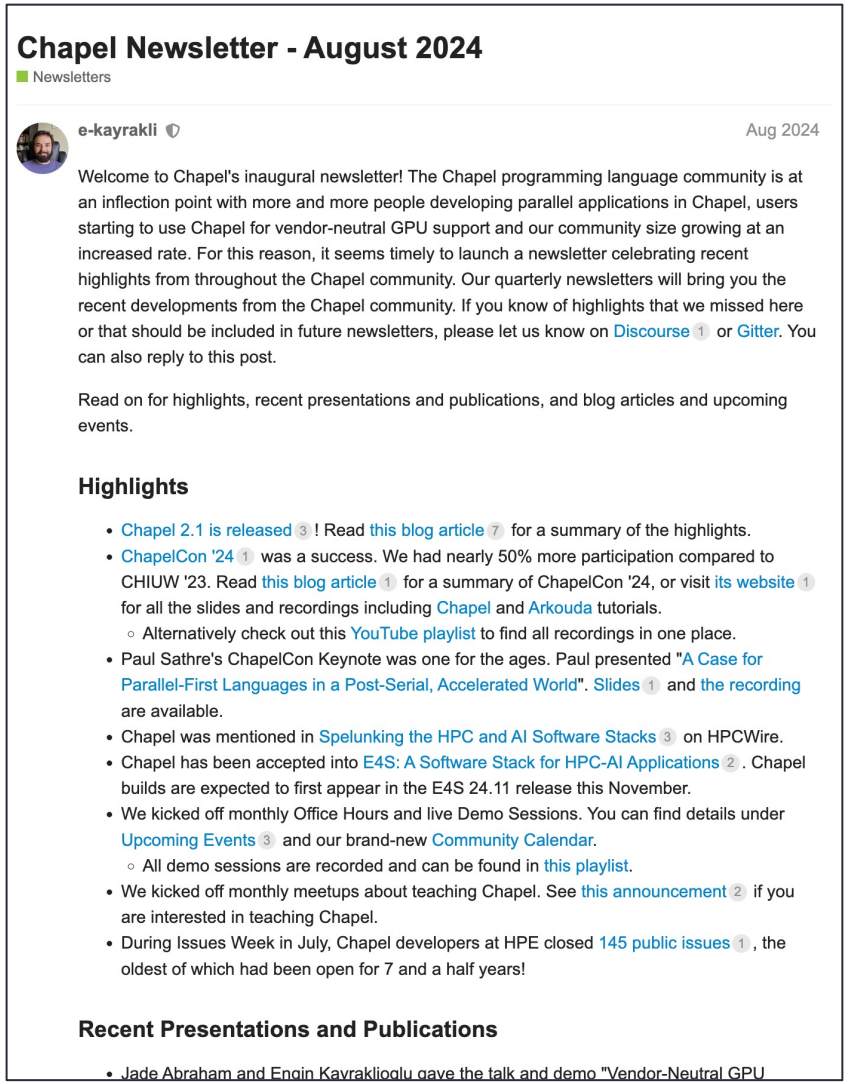



# A New Chapel Newsletter!

We launched a new, quarterly newsletter in August 2024

- Browse / subscribe through Discourse:
  - https://chapel.discourse.group/c/newsletters



## Chapel Newsletter - August 2024

Newsletters

e-kayrakli Aug 2024

Welcome to Chapel's inaugural newsletter! The Chapel programming language community is at an inflection point with more and more people developing parallel applications in Chapel, users starting to use Chapel for vendor-neutral GPU support and our community size growing at an increased rate. For this reason, it seems timely to launch a newsletter celebrating recent highlights from throughout the Chapel community. Our quarterly newsletters will bring you the recent developments from the Chapel community. If you know of highlights that we missed here or that should be included in future newsletters, please let us know on Discourse 1 or Gitter. You can also reply to this post.

Read on for highlights, recent presentations and publications, and blog articles and upcoming events.

### Highlights

- Chapel 2.1 is released 3 ! Read this blog article 7 for a summary of the highlights.
- ChapelCon '24 1 was a success. We had nearly 50% more participation compared to CHIUW '23. Read this blog article 1 for a summary of ChapelCon '24, or visit its website 1 for all the slides and recordings including Chapel and Arkouda tutorials.
  - Alternatively check out this YouTube playlist to find all recordings in one place.
- Paul Sathre's ChapelCon Keynote was one for the ages. Paul presented "A Case for Parallel-First Languages in a Post-Serial, Accelerated World". Slides 1 and the recording are available.
- Chapel was mentioned in Spelunking the HPC and AI Software Stacks 3 on HPCWire.
- Chapel has been accepted into E4S: A Software Stack for HPC-AI Applications 2 . Chapel builds are expected to first appear in the E4S 24.11 release this November.
- We kicked off monthly Office Hours and live Demo Sessions. You can find details under Upcoming Events 3 and our brand-new Community Calendar.
  - All demo sessions are recorded and can be found in this playlist.
- We kicked off monthly meetups about teaching Chapel. See this announcement 2 if you are interested in teaching Chapel.
- During Issues Week in July, Chapel developers at HPE closed 145 public issues 1 , the oldest of which had been open for 7 and a half years!

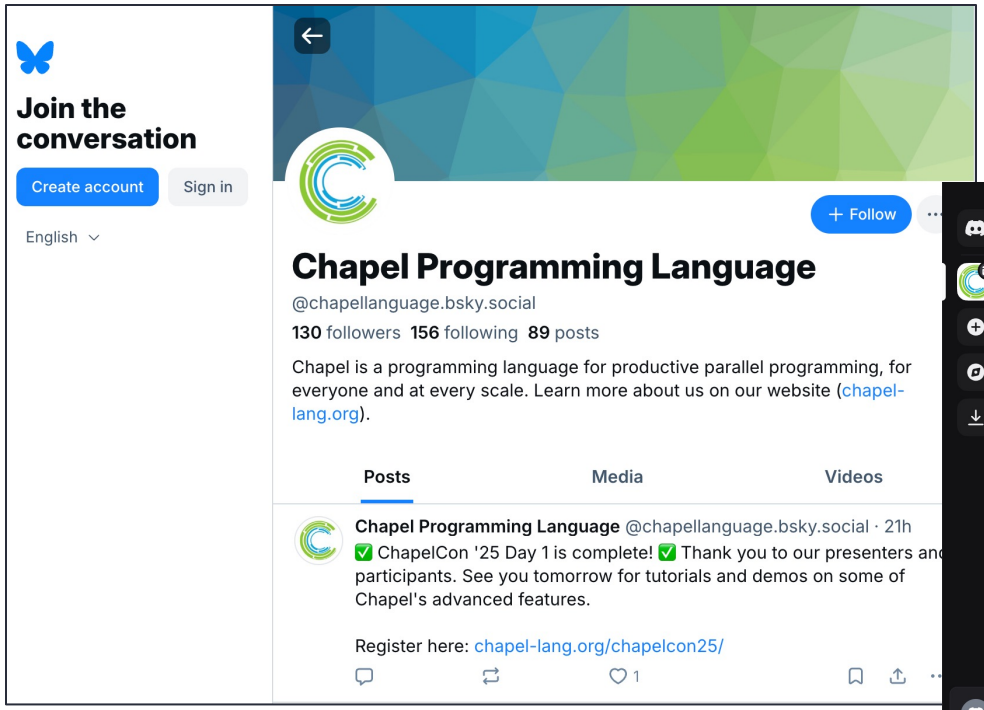
### Recent Presentations and Publications

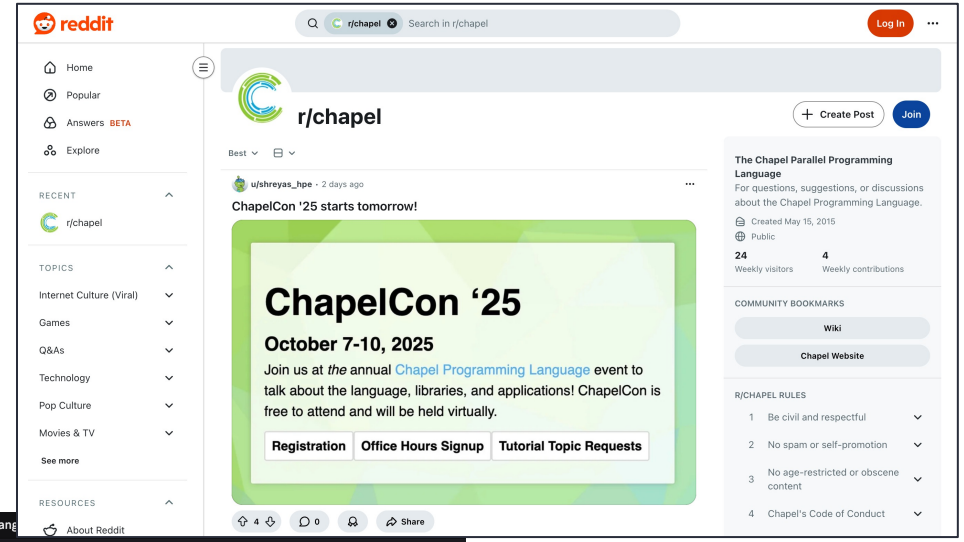
- Jade Abraham and Engin Kayraklioglu gave the talk and demo "Vendor-Neutral GPU

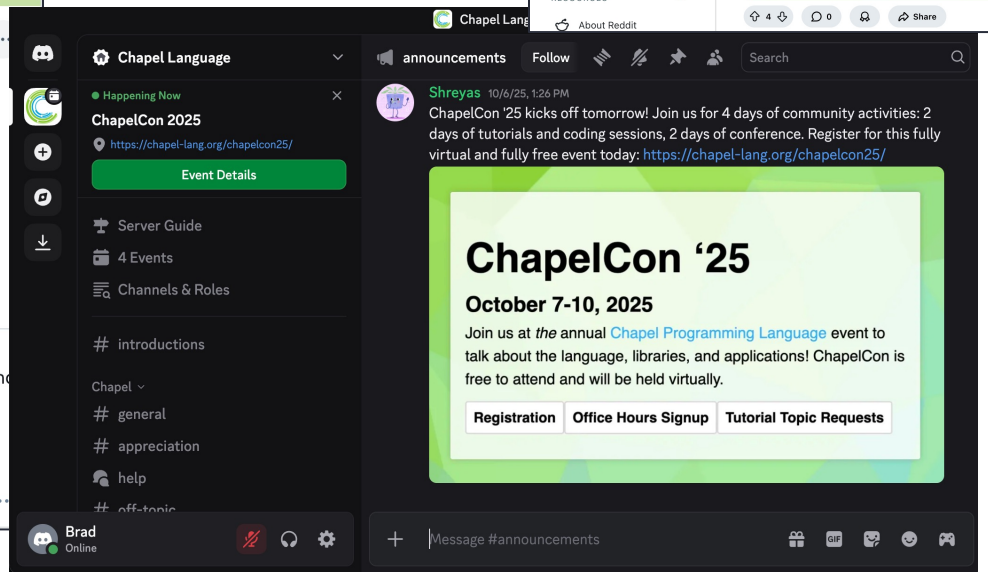
# New Social Media / Community Forums

— Added new channels to previous (LinkedIn, X, Mastodon, Facebook):
  - **Reddit:** Began posting regularly, Aug 2024
  - **Discord:** Launched server, Nov 2024
  - **BlueSky:** Launched account, Jan 2025

— Averaging 2–3 social media posts / week, on average



# “7 Questions with Chapel Users”

We launched a new *7 Questions with Chapel Users* interview blog series, capturing users’ perspectives

Chapel Language Blog

About Chapel Website Featured Series Tags Authors All Posts

**7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel**

Posted on September 17, 2024.

Tags: Computational Fluid Dynamics | User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

**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 Kayraklioglu

**7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel**

Posted on October 15, 2024.

Tags: User Experiences | Interviews | Data Analysis | Earth Sciences | Computational Fluid Dynamics

By: Engin Kayraklioglu, Brad Chamberlain

**7 Questions for David Bader: Graph Analytics at Scale with Arkouda and Chapel**

Posted on November 6, 2024.

Tags: User Experiences | Interviews | Graph Analytics | Arkouda

By: Engin Kayraklioglu, Brad Chamberlain

**7 Questions for Bill Reus: Interactive Supercomputing with Chapel for Cybersecurity**

Posted on February 12, 2025.

Tags: User Experiences | Interviews | Data Analysis | Arkouda

By: Engin Kayraklioglu, Brad Chamberlain

**7 Questions for Tiago Carneiro and Guillaume Helbecque: Combinatorial Optimization in Chapel**

Posted on July 30, 2025.

Tags: User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

**7 Questions for Marjan Asgari: Optimizing Hydrological Models with Chapel**

Posted on September 15, 2025.

Tags: User Experiences | Interviews | Earth Sciences

By: Engin Kayraklioglu, Brad Chamberlain

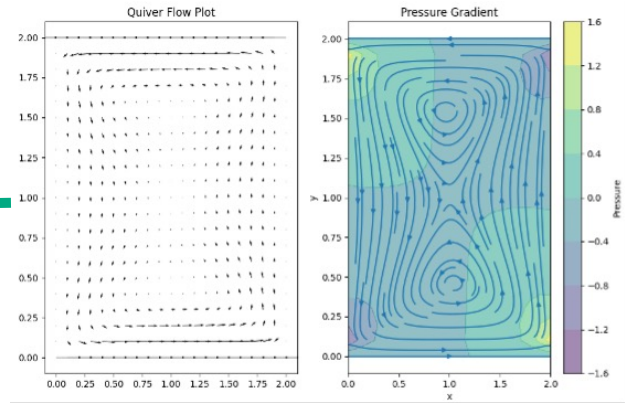
**Your Chapel story here?**
(contact us if interested)

# New Blog Articles

— 33 new articles in the 70 weeks since ChapelCon '24

— **New series:**

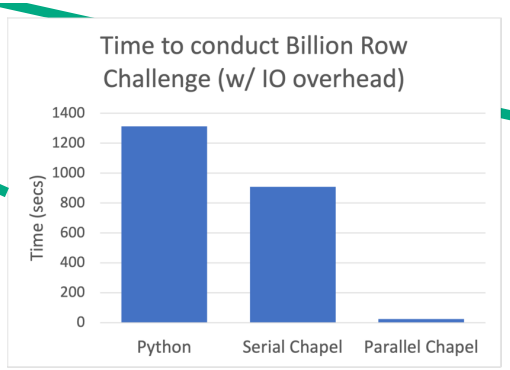
- 7 Questions with Chapel Users
- 4-part Navier-Stokes in Chapel
- 10 Myths of Productive Scalable Programming Languages (Redux)

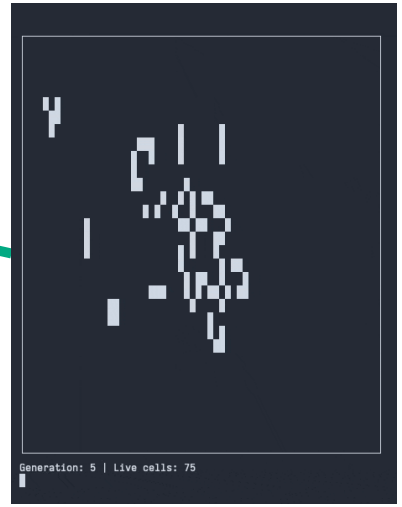



— **Standout standalone articles** (not the actual titles):

- **Memory Safety** in Chapel (relative to Rust, Python, C, C++)
- Using **GenAI** to write / refactor Chapel
- The **1 Billion Row Challenge** in Chapel
- Using Chapel on your **Windows Gaming GPU**
- **Chapel/Fortran** Interop in an ocean model
- **Hyperparameter Optimization** in Chapel
- Using **Chapel's Python bindings** for tooling
- Report from **SC24**

| Error | C | C++ | Rust | Python | Chapel |
|---|---|---|---|---|---|
| Variable Not Initialized | ❌ | ❌ | ✅ | ✅ | ✅ |
| Mishandling Strings | ❌ | ⚠️ | ✅ | ✅ | ✅ |
| Use-After-Free | ❌ | ⚠️ | ⚠️ | ✅ | ⚠️ |
| Out-of-Bounds Array Access | ❌ | ❌ | ✅ | ✅ | ⚠️ |



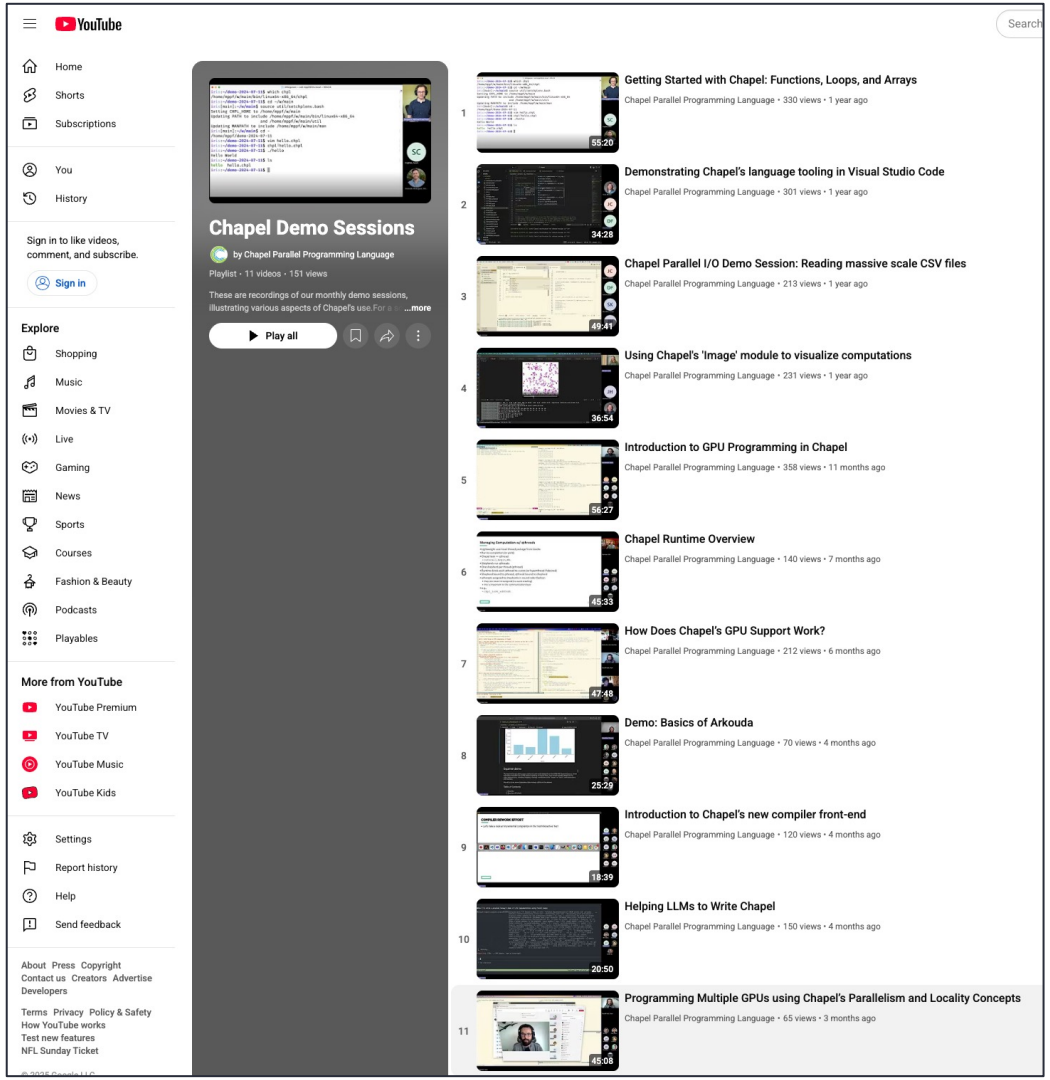



# Other New Community Events

— new ~monthly **Chapel Demo**
  - launched July 2024
  - archived on YouTube

— new **Teaching in Chapel** monthly meetup
  - launched by Michelle Strout in Aug 2024
  - currently led by Alex Razoumov
  - Chapel Examples and Teaching Materials GitHub repo

— monthly **Office Hours**
  - (now folded into other weekly meetings)



# Congratulations to Dr. Oliver Alvarado Rodriguez!

**The latest Ph.D. focusing on a Chapel-related topic!**

*On the Design of a Framework for Large-Scale Exploratory Graph Analytics*

(advised by Professor David Bader, NJIT)

- Dissertation available online
- Now working with us at HPE on the Advanced Programming team

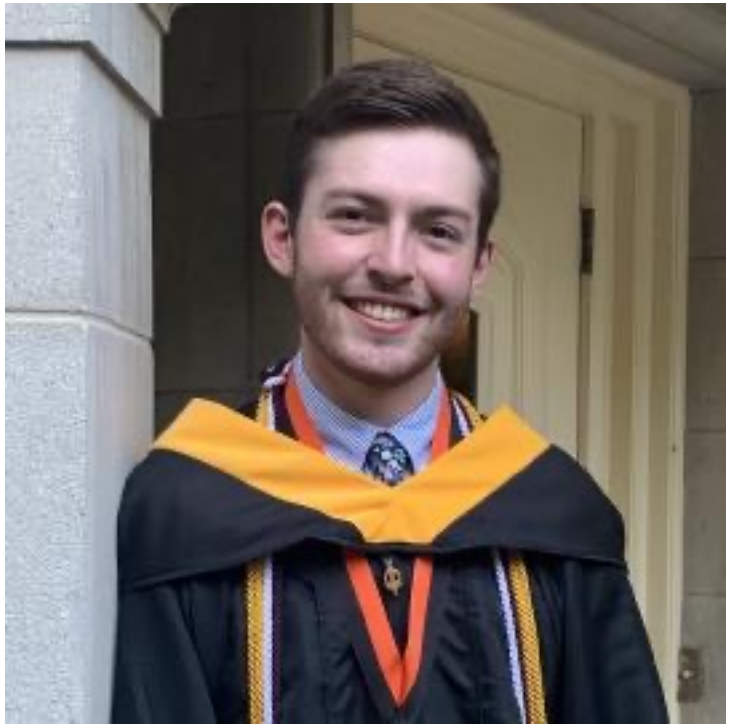
ON THE DESIGN OF A FRAMEWORK FOR LARGE-SCALE
EXPLORATORY GRAPH ANALYTICS

by
Oliver Andres Alvarado Rodriguez

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2025



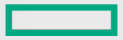
# Plus: Sooooo many community talks, papers, and events

A few highlights that stand out:

- **Paul Sathre's ChapelCon '24 keynote**, "A Case for Parallel Languages in a Post-Serial, Accelerated World"
- Josh Milthorpe's HCW talk, "Performance Portability of the Chapel Language on Heterogeneous Architectures"
- Guillaume Helbecque's IPDPS talk, "GPU-Accelerated Tree-Search in Chapel: Comparing Against CUDA and HIP on Nvidia and AMD GPUs"
- Eric Laurendeau's PAW-ATM distinguished talk, "A case study for using Chapel within the global aerospace industry"
- Jeremiah Corrado's PANGEO demo, "Arkouda as an XArray Backend for HPC"
- My HPCwire interview, "What's New with Chapel? Nine Questions for the Development Team"
- Alex Razoumov's webinars, like "GPU Computing with Chapel"
- Tiago Carneiro's Euro-Par talk, "Investigating Portability in Chapel for Tree-Based Optimization on GPU-Powered Clusters"
- Michelle Strout's CCDSC talk, "Real Applications, Real Fast in Chapel"
- Mohammad Dindoost and Garrett Gonzalez-Rivas' talks at HPEC, "VF2-PS: Parallel and Scalable Subgraph Monomorphism in Arachne" and "A Deployment Tool for Large Scale Graph Analytics Framework Arachne"
- Engin Kayraklioglu, Éric Laurendeau, and Karim Zayni's joint talk at NASA, "High-Performance, Productive Programming using Chapel with Examples from the CFD Solver CHAMPS"
- Ivan Tagliaferro de Oliveira Tezoto's IPDPS poster, "Performance and Portability in Multi-GPU Branch-and-Bound: Chapel versus CUDA and HIP for Tree-Based Optimization"
- Bokyeong Yoon's HIPS presentation "Exploring Communication Anomalies in Chapel"
- Luca Ferranti's JuliaCon BoF, "Chapel ❤️ Julia"
- My HIPS keynote, "Reflections on 30 years of HPC programming: So many hardware advances, so little adoption of new languages"

(See the newsletters for a far more complete list)

# Chapel and HPSF



# Chapel's Inception

**Conversation with Burton Smith, ~Oct 2002** (paraphrased)

- Chief Scientist of Cray Inc., co-founder of Tera
- PI of Cray's HPCS program, Cascade

— Me: "To improve user productivity, we should create a new language!"

— Burton: "No, I fear a language developed by a single vendor will not be successful"

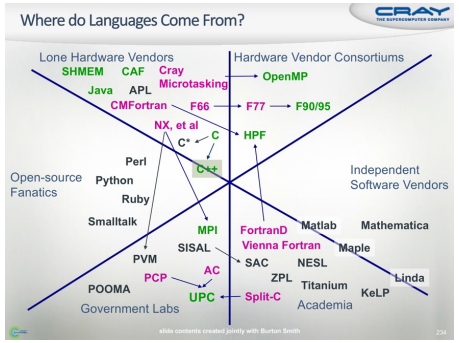
— Me: "OK… :'(" [who am I to argue with Burton?]

**~Nov 2002:**

— Me: "But wait, several important languages started with a single vendor…"

— Burton: "Good point, let's do a thought experiment…"

— [a few days later…]




— Burton: "OK. And note that successful single-vendor languages typically made a jump to community governance…"

# What is HPSF?

**HPSF** = High Performance Software Foundation

— a neutral hub for open-source HPC software

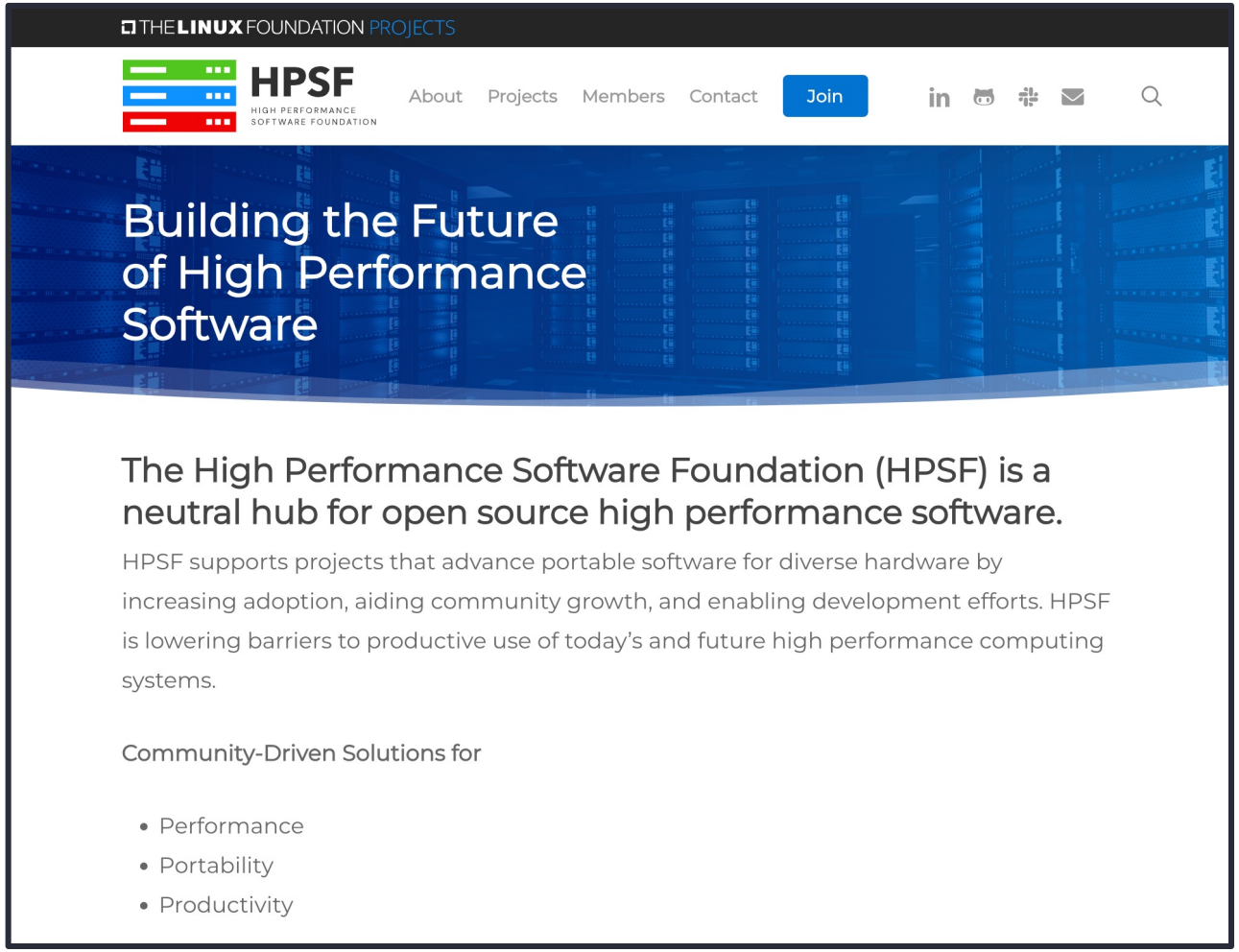
— a Linux Foundation project

— **mission:** "to constantly improve the quality and open availability of software for HPC through open collaboration", focusing on:

- performance
- portability
- productivity

— **goals for member projects:**

- increase adoption
- aid community growth
- enable development efforts



# Synergies Between HPSF's Goals and Chapel

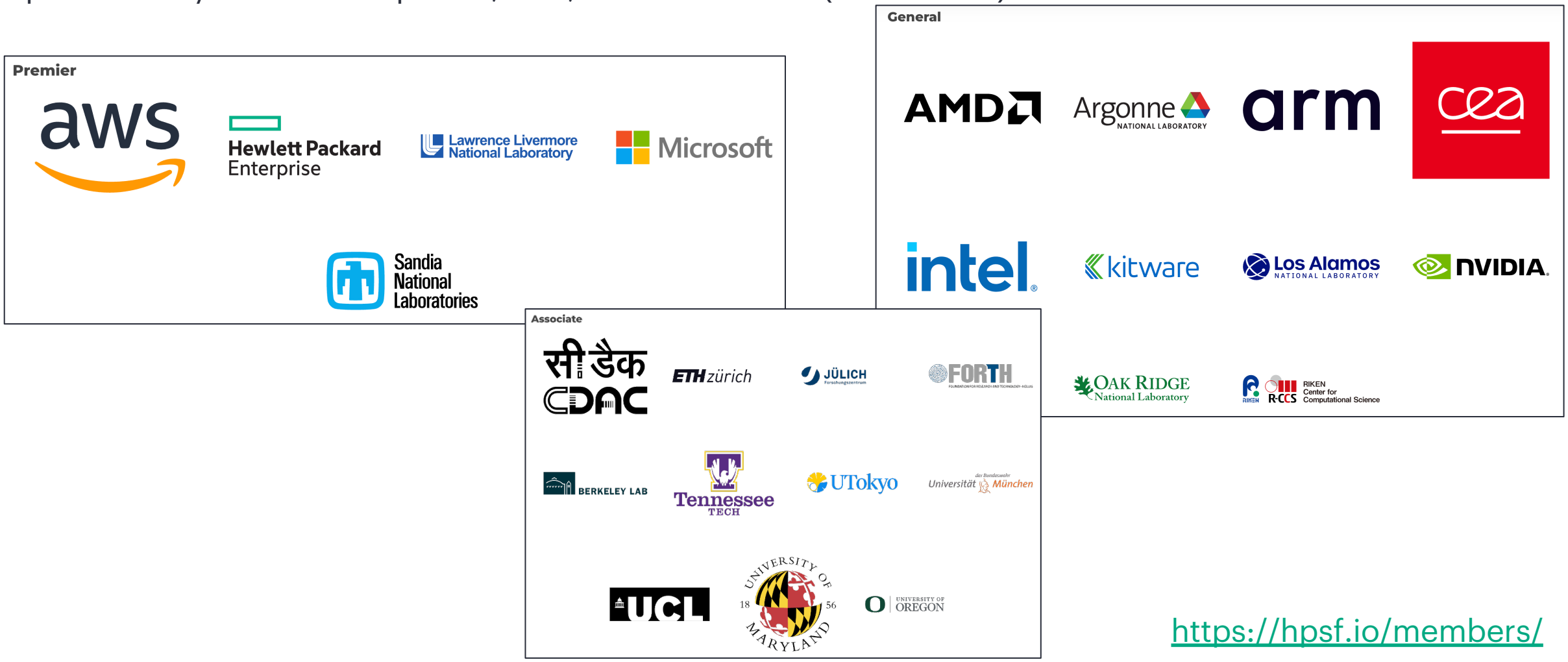
| | |
|---|---|
| **Lowering barriers to using HPC** | • Chapel helps real users write real applications<br>  • Many are writing HPC code for the first time<br>  • Others are HPC experts, working more quickly than they could've<br>  • Others are simply leveraging their desktop multicore CPUs + GPUs |
| **Aiding HPC community growth** | |
| **Enabling HPC development efforts** | |
| **Portable software for diverse HW** | • Chapel currently supports most any HPC, desktop, or cloud system<br>• Its language design and SW architecture support porting to others |
| **Performance and productivity** | • Chapel performance often matches or beats conventional HPC models<br>• Code is almost always shorter and easier to read/write/maintain |

## Motivations for Applying:

- Have always intended to move governance of project into the community
  - Goal: shed the "single-vendor" stigma
- A chance to network with other open-source HPC projects and share best practices
  - Share our experience that may be useful to other projects
  - We have lots to learn as well
- Anticipated benefits to Chapel's visibility and stature

# Who sponsors HPSF?

Sponsored by various companies, labs, and universities (“members”):




https://hpsf.io/members/

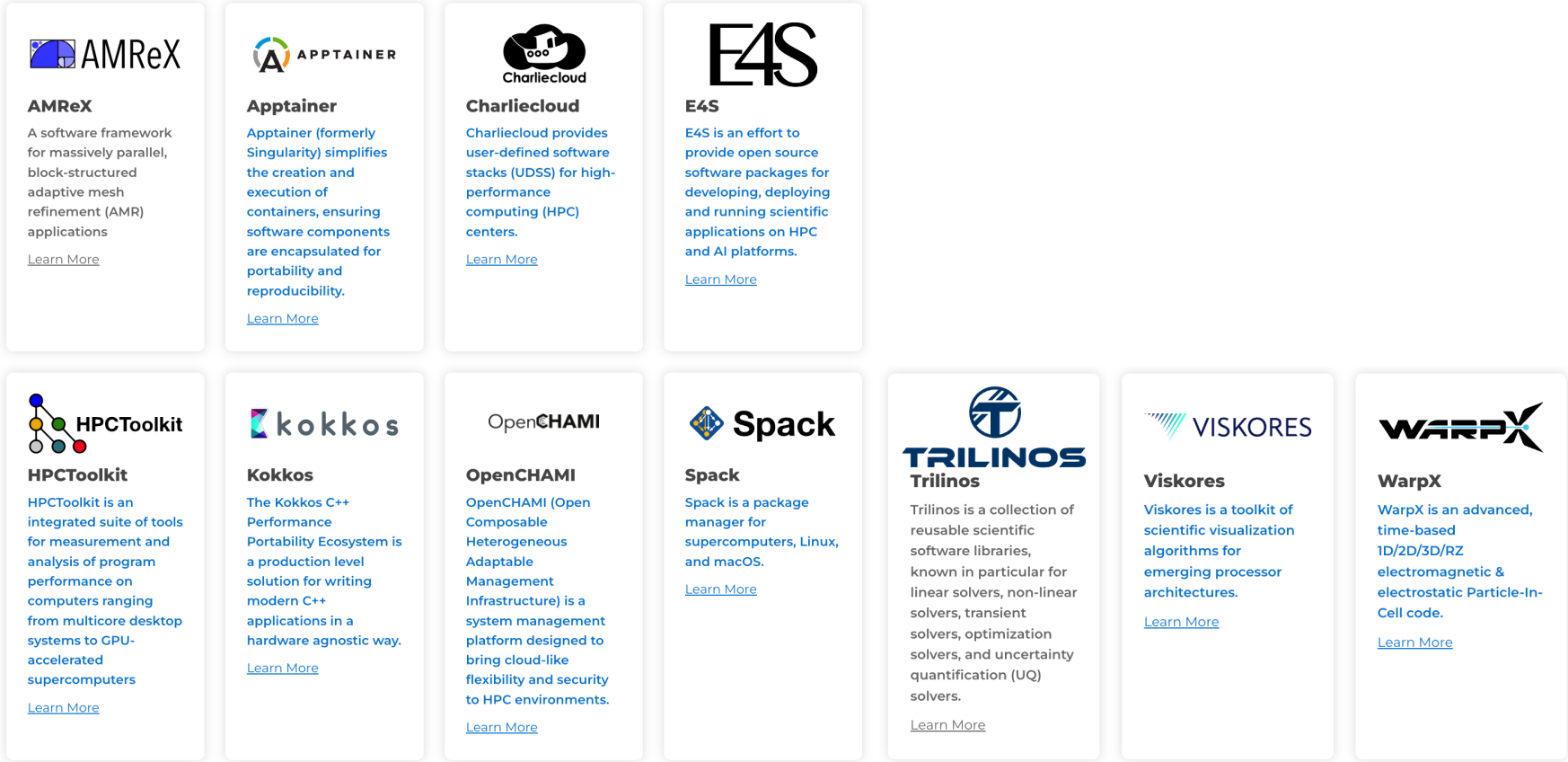
# What projects are involved?

Founding projects are:

- AMReX
- Apptainer
- Charliecloud
- E4S
- HPCToolkit
- Kokkos
- Spack
- Trilinos
- Viskores
- WarpX

And since then:

- OpenCHAMI




https://hpsf.io/projects/

# Chapel's HPSF Timeline

**2023:**

- **Nov 13 @ SC23:** Public statements of intention to form HPSF

**2024:**

- **May 13:** Linux Foundation announces launch of HPSF
- **Sept 3:** First applications are submitted by founding projects
- **Sept 19:** Chapel encouraged to apply after inquiring about the possibility
- **Oct 1:** Submitted our application

**2025:**

- **Jan 9:** Presented our application to the HPSF Technical Advisory Committee (TAC)
- **Jan 23:** Learned we were accepted
  - Kicked off legal processes at both HPE and LF
- **April:** Made weekly project meetings and chapel-www repo public
- **May 22:** Made weekly deep-dive meetings public
- **July:** Made chapel-blog repo public: https://github.com/chapel-lang/chapel-blog
- **Aug 26–Sept 2:** Formed the initial technical steering committee (TSC)
- **Sept 16–18:** Finalized technical charter and held first TSC meeting, approving it
- **Sept 25:** Signed the paperwork transferring Chapel name and accounts to HPSF/LF

# Chapel and HPSF: What's Next?

**Create a new logo:**

- Our traditional logo was not part of the transfer to the Linux Foundation
- See TSC issue #17 for details
- Quickly crowd-sourcing designs?

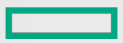

**Do a big announcement, jointly with HPSF/LF and HPE**

- In time for SC25?

**Thereafter, open up additional aspects of the project**

- Determine application process for prospective new Technical Steering Committee members
- Determine how to add new project committers
- …

# Excerpts from “Reflections on 30 Years of HPC Programming”



# 30 Years Ago vs. Now: Top HPC Systems

Top 5 systems in the Top500, June 1995:

- **Cores:** 80–3680 cores
- **Rmax:** ~98.9–170 GFlop/s
- **Systems:** Fujitsu, Intel Paragon XP/S, Cray T3D
- **Networks:** crossbar, mesh, 3D torus

TOP500 LIST - JUNE 1995

$R_{max}$ and $R_{peak}$ values are in GFlop/s. For more details about other fields, check the TOP500 description.

$R_{peak}$ values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

← 1-100 | 101-200 | 201-300 | 301-400 | 401-500 →

| Rank | System | Cores | Rmax (GFlop/s) | Rpeak (GFlop/s) | Power (kW) |
|---|---|---|---|---|---|
| 1 | Numerical Wind Tunnel, Fujitsu<br>National Aerospace Laboratory of Japan<br>Japan | 140 | 170.00 | 235.79 | |
| 2 | XP/S140, Intel<br>Sandia National Laboratories<br>United States | 3,680 | 143.40 | 184.00 | |
| 3 | XP/S-MP 150, Intel<br>DOE/SC/Oak Ridge National Laboratory<br>United States | 3,072 | 127.10 | 154.00 | |
| 4 | T3D MC1024-8, Cray/HPE<br>Government<br>United States | 1,024 | 100.50 | 153.60 | |
| 5 | VPP500/80, Fujitsu<br>National Lab. for High Energy Physics<br>Japan | 80 | 98.90 | 128.00 | |

Top 5 systems in the Top 500, June 2025:

- **Cores:** 2,073,600–11,039,616 **(~563x–138,000x)**
- **Rmax:** ~477.9–1742.0 PFlop/s **(~2,810,000x–17,600,000x)**
- **Systems:** HPE Cray EX, Eviden Bullsequana, Microsoft Azure
- **Networks:** Slingshot-11, InfiniBand NDR

TOP500 LIST - JUNE 2025

$R_{max}$ and $R_{peak}$ values are in PFlop/s. For more details about other fields, check the TOP500 description.

$R_{peak}$ values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

← 1-100 | 101-200 | 201-300 | 301-400 | 401-500 →

| Rank | System | Cores | Rmax (PFlop/s) | Rpeak (PFlop/s) | Power (kW) |
|---|---|---|---|---|---|
| 1 | **El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE<br>DOE/NNSA/LLNL<br>United States | 11,039,616 | 1,742.00 | 2,746.38 | 29,581 |
| 2 | **Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE<br>DOE/SC/Oak Ridge National Laboratory<br>United States | 9,066,176 | 1,353.00 | 2,055.72 | 24,607 |
| 3 | **Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel<br>DOE/SC/Argonne National Laboratory<br>United States | 9,264,128 | 1,012.00 | 1,980.01 | 38,698 |
| 4 | **JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN<br>EuroHPC/FZJ<br>Germany | 4,801,344 | 793.40 | 930.00 | 13,088 |
| 5 | **Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure<br>Microsoft Azure<br>United States | 2,073,600 | 561.20 | 846.84 | |

# 30 Years Ago vs. Now: Top HPC Systems

Top 5 systems in the Top500, June 1995:

- **Cores:** 80–3680 cores
- **Rmax:** ~98.9–170 GFlop/s
- **Systems:** Fujitsu, Intel Paragon XP/S, Cray T3D
- **Networks:** crossbar, mesh, 3D torus

**HPC HW has become far more capable…**

Top 5 systems in the Top 500, June 2025:

- **Cores:** 2,073,600–11,039,616 **(~563x–138,000x)**
- **Rmax:** ~477.9–1742.0 PFlop/s **(~2,810,000x–17,600,000x)**
- **Systems:** HPE Cray EX, Eviden Bullsequana, Microsoft Azure
- **Networks:** Slingshot-11, InfiniBand NDR

TOP500 LIST - JUNE 1995

$R_{max}$ and $R_{peak}$ values are in GFlop/s. For more details about other fields, check the TOP500 description.

$R_{peak}$ values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

1-100 | 101-200 | 201-300 | 301-400 | 401-500

| Rank | System | | | |
|---|---|---|---|---|
| 1 | Numerical Wind Tunnel, Fujitsu<br>National Aerospace Laboratory of Japan<br>Japan | | | |
| 2 | XP/S140, Intel<br>Sandia National Laboratories<br>United States | | | |
| 3 | XP/S-MP 150, Intel<br>DOE/SC/Oak Ridge National Laboratory<br>United States | | | |
| 4 | T3D MC1024-8, Cray/HPE<br>Government<br>United States | | | |
| 5 | VPP500/80, Fujitsu<br>National Lab. for High Energy Physics<br>Japan | 80 | 98.90 | 128.00 |

**And complex!**

- **commodity vector processors**
- **multicore processors**
- **multi-socket compute nodes**
- **NUMA compute node architectures**
- **high-radix, low-diameter interconnects**
- **GPU computing**

**(Often in ways that hurt programmability)**

TOP500 LIST - JUNE 2025

$R_{max}$ and $R_{peak}$ values are in PFlop/s. For more details about other fields, check the TOP500 description.

$R_{peak}$ values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

1-100 | 101-200 | 201-300 | 301-400 | 401-500

| System | Cores | Rmax (PFlop/s) | Rpeak (PFlop/s) | Power (kW) |
|---|---|---|---|---|
| …an - HPE Cray EX255a, AMD 4th Gen EPYC 24C …MD Instinct MI300A, Slingshot-11, TOSS, HPE …L NL | 11,039,616 | 1,742.00 | 2,746.38 | 29,581 |
| …235a, AMD Optimized 3rd …z, AMD Instinct MI250X, …PE …oratory | 9,066,176 | 1,353.00 | 2,055.72 | 24,607 |
| …xascale Compute Blade, …GHz, Intel Data Center GPU …ional Laboratory | 9,264,128 | 1,012.00 | 1,980.01 | 38,698 |
| …**Booster** - BullSequana XH3000, GH Superchip …z, NVIDIA GH200 Superchip, Quad-Rail NVIDIA …Gand NDR200, RedHat Enterprise Linux, EVIDEN …roHPC/FZJ<br>Germany | 4,801,344 | 793.40 | 930.00 | 13,088 |
| **Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure<br>Microsoft Azure<br>United States | 2,073,600 | 561.20 | 846.84 | |

# 30 Years Ago vs. Now: Top HPC Systems and Programming Notations

Top 5 systems in the Top500, June 1995:

- **Cores:** 80–3680 cores
- **Rmax:** ~98.9–170 GFlop/s
- **Systems:** Fujitsu, Intel Paragon XP/S, Cray T3D
- **Networks:** crossbar, mesh, 3D torus

**HPC HW has become far more capable…**

Top 5 systems in the Top 500, June 2025:

- **Cores:** 2,073,600–11,039,616 **(~563x–138,000x)**
- **Rmax:** ~477.9–1742.0 PFlop/s **(~2,810,000x–17,600,000x)**
- **Systems:** HPE Cray EX, Eviden Bullsequana, Microsoft Azure
- **Networks:** Slingshot-11, InfiniBand NDR

Broadly-adopted HPC programming notations:

- **Languages:** C, C++, Fortran
- **Inter-node:** MPI, SHMEM
- **Intra-node:** vendor-specific pragmas & intrinsics
  - OpenMP on the horizon: 1997
- **Scripting:** Perl, [[t]c]sh, Tcl/TK

Broadly-adopted HPC programming notations:

- **Languages:** C, C++, Fortran
- **Inter-node:** MPI, SHMEM
- **Intra-node:** OpenMP, vendor-specific pragmas & intrinsics
- **GPUs:** CUDA, HIP, SYCL, Kokkos, OpenMP, OpenACC, …
- **Scripting:** Python, bash

# 30 Years Ago vs. Now: Top HPC Systems and Programming Notations

Top 5 systems in the Top500, June 1995:

- **Cores:** 80–3680 cores
- **Rmax:** ~98.9–170 GFlop/s
- **Systems:** Fujitsu, Intel Paragon XP/S, Cray T3D
- **Networks:** crossbar, mesh, 3D torus

**HPC HW has become far more capable…**

Top 5 systems in the Top 500, June 2025:

- **Cores:** 2,073,600–11,039,616 **(~563x–138,000x)**
- **Rmax:** ~477.9–1742.0 PFlop/s **(~2,810,000x–17,600,000x)**
- **Systems:** HPE Cray EX, Eviden Bullsequana, Microsoft Azure
- **Networks:** Slingshot-11, InfiniBand NDR

Broadly-adopted HPC programming notations:

- **Languages:** C, C++, Fortran
- **Inter-node:** MPI, SHMEM
- **Intra-node:** vendor-specific pragmas & intrinsics
  - OpenMP on the horizon: 1997
- **Scripting:** Perl, [[t]c]sh, Tcl/TK

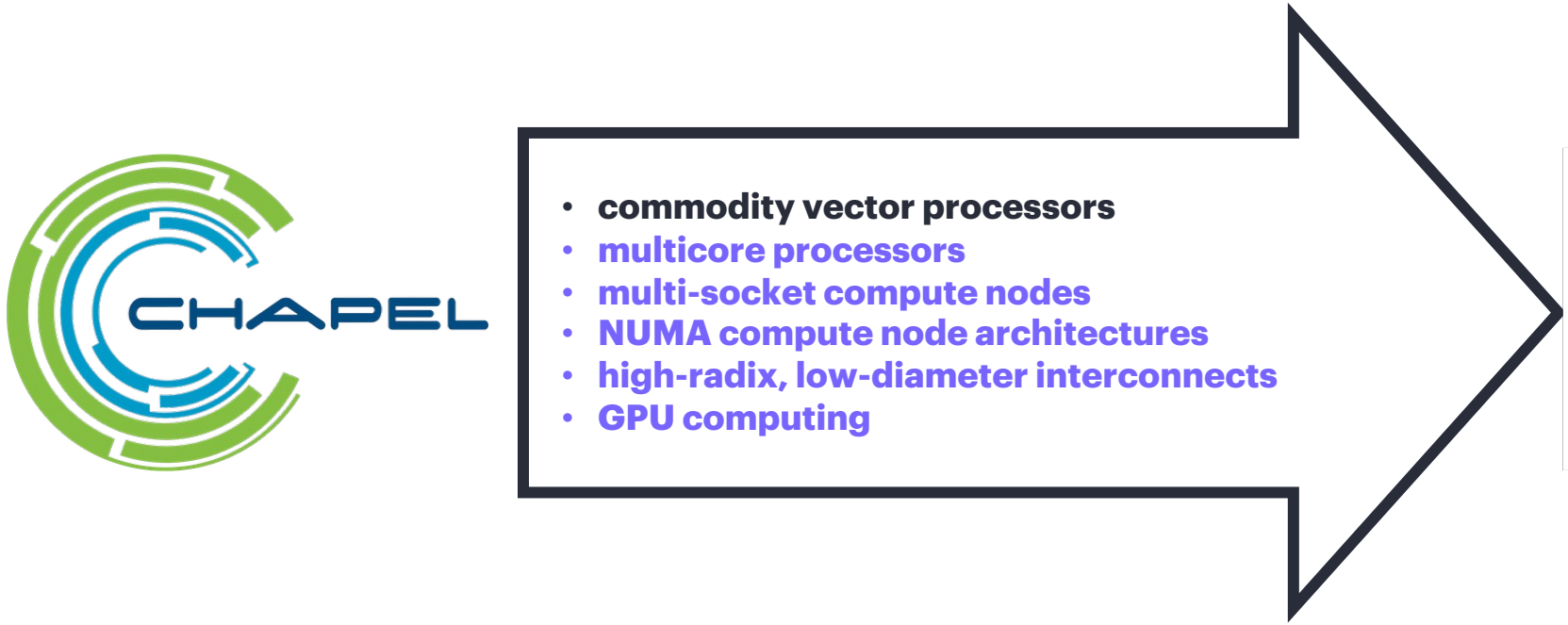
**…while HPC notations have largely stayed the same, modulo GPU computing**

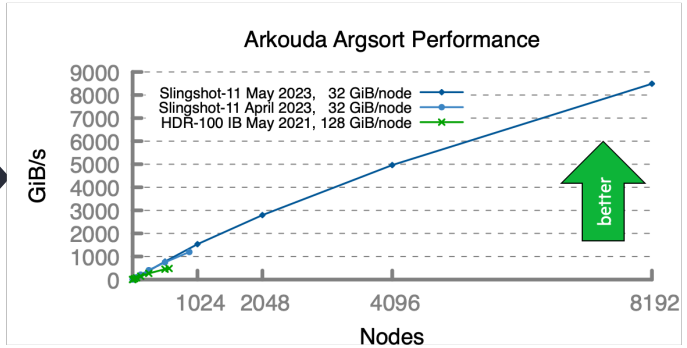
Broadly-adopted HPC programming notations:

- **Languages:** C, C++, Fortran
- **Inter-node:** MPI, SHMEM
- **Intra-node:** OpenMP, vendor-specific pragmas & intrinsics
- **GPUs:** CUDA, HIP, SYCL, Kokkos, OpenMP, OpenACC, …
- **Scripting:** Python, bash

# Chapel's adaptable persistence

Chapel predates all of the architectural changes mentioned previously, apart from commodity vectors

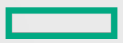



Yet it supports all of these HW features

- Using essentially the same language features as ~20 years ago
- How? By focusing on expressing parallelism and locality independently from HW mechanisms

# Chapel in the age of AI



# AI, HPC, and Languages

**Q: AI can program now*. Do languages like Chapel still matter?**

**My answer is a resounding "yes"…**

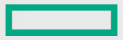
- To say we no longer need good programming languages and compilers in the age of AI is like saying we no longer need to invest in roads, automobile manufacturing, fuel efficiency, safety, and traditional driving skills in an age of self-driving cars.

**Value proposition:**

- humans will still program
- higher-level, clearer languages should enable greater AI successes
- when users need to look under the hood, the more comprehensible the code is, the better

(* = your mileage may vary)

# Wrapping Up



**A huge amount of progress has been made since ChapelCon '24**

**We're looking forward to what the year ahead holds!**

# Thank You

@ChapelLanguage