Hewlett Packard Enterprise

The Chapel Parallel Programming Language and its Ecosystem

Engin Kayraklioglu
engin@hpe.com
linkedin.com/in/engink

Jade Abraham
jade.abraham@hpe.com
linkedin.com/in/jabraham17

October 4, 2024

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-lang.org

What is Chapel?

Chapel works everywhere

- you can develop on your laptop and have the code scale on a supercomputer
- GPUs can be targeted in a vendor-neutral way
- runs on Linux laptops/clusters, Cray systems, MacOS, WSL, AWS, Raspberry Pi
- shown to scale on Cray networks (Slingshot, Aries), InfiniBand, RDMA-Ethernet

Chapel makes distributed/shared memory parallel programming easy

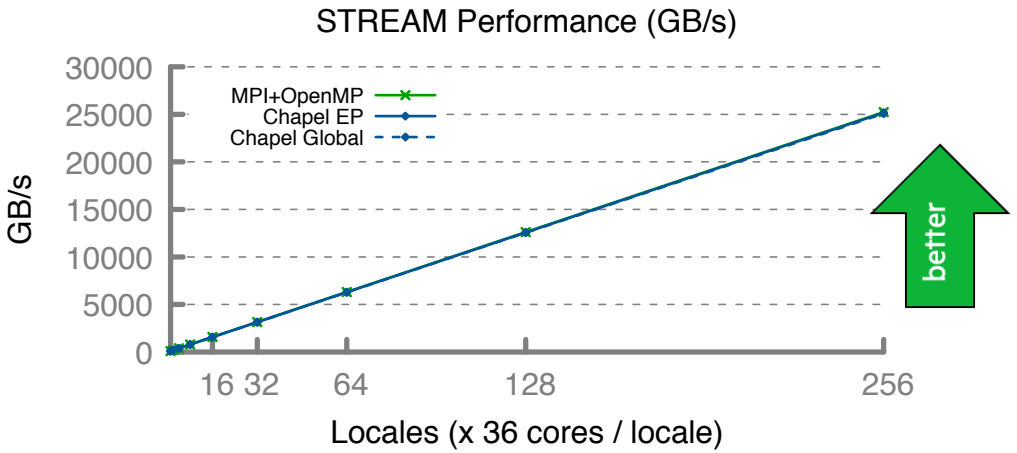
- data-parallel, locality-aware loops,
- ability to move execution and allocation to remote nodes,
- distributed arrays and bulk array operations
- different types of parallelism can be expressed with the same language features

HPCC Stream Triad and RA: C + MPI +OpenMP vs. Chapel

STREAM TRIAD: C + MPI + OPENMP

```
#include <hpcc.h>
#ifdef _OPENMP
#include <omp.h>
#endif

static int VectorSize;
static double *a, *b, *c;

int HPCC_StarStream(HPCC_Params *params) {
  int myRank, commSize;
  int rv, errCount;
  MPI_Comm comm = MPI_COMM_WORLD;

  MPI_Comm_size( comm, &commSize );
  MPI_Comm_rank( comm, &myRank );

  rv = HPCC_Stream( params, 0 == myRank);
  MPI_Reduce( &rv, &errCount, 1, MPI_INT, MPI_SUM, 0, comm );

  return errCount;
}

int HPCC_Stream(HPCC_Params *params, int doIO) {
  register int j;
  double  scalar;

  VectorSize = HPCC_LocalVectorSize( params, 3, sizeof(double), 0 );

  a = HPCC_XMALLOC( double, VectorSize );
  b = HPCC_XMALLOC( double, VectorSize );
  c = HPCC_XMALLOC( double, VectorSize );
```

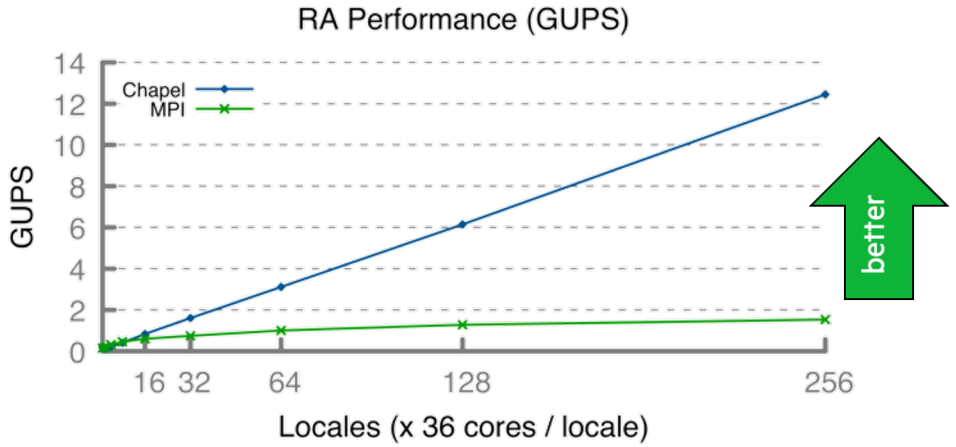
```
use BlockDist;

config const n = 1_000_000,
             alpha = 0.01;
const Dom = blockDist.createDomain({1..n});
var A, B, C: [Dom] real;

B = 2.0;
C = 1.0;

A = B + alpha * C;
```



HPCC RA: MPI KERNEL

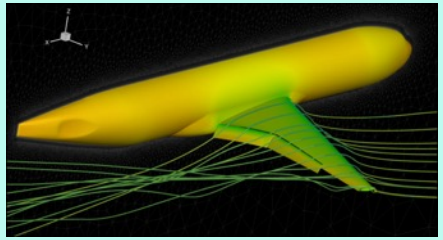
```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *   for (i=0; i<NUPDATE; i++){
 *     Ran = (Ran << 1) ^ ((s64int) Ran < 0) ? POLY : 0;
 *     Table[Ran & (TABSIZE-1)] ^= Ran;
 *   }
 */

MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
          MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
while (i < SendCnt) {
  /* receive messages */
  do {
    MPI_Test(&inreq, &have_done, &status);
    if (have_done) {
      if (status.MPI_TAG == UPDATE_TAG) {
        MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
        bufferBase = 0;
        for (j=0; j < recvUpdates; j ++) {
          inmsg = LocalRecvBuffer[bufferBase+j];
          LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                        tparams.GlobalStartMyProc;
          HPCC_Table[LocalOffset] ^= inmsg;
        }
      } else if (status.MPI_TAG == FINISHED_TAG) {
        NumberReceiving--;
      } else
        MPI_Abort( MPI_COMM_WORLD, -1 );
      MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
    }
  } while (have_done && NumberReceiving > 0);
  if (pendingUpdates < maxPendingUpdates) {
    Ran = (Ran << 1) ^ (s64Int) Ran < ZERO64B ? POLY : ZERO64B;
    GlobalOffset = Ran & (tparams.TableSize-1);
    if ( GlobalOffset < tparams.Top)
      WhichPe = ( GlobalOffset / (tparams.MinLocalTableSize + 1) );
    else
      WhichPe = ( (GlobalOffset - tparams.Remainder) /
                  tparams.MinLocalTableSize );
    if (WhichPe == tparams.MyProc) {
      LocalOffset = (Ran & (tparams.TableSize - 1)) -
                    tparams.GlobalStartMyProc;
      HPCC_Table[LocalOffset] ^= Ran;
    } else {
      HPCC_InsertUpdate(Ran, WhichPe, Buckets);
      pendingUpdates++;
    }
    i++;
  }
  else {
    MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
    if (have_done) {
      outreq = MPI_REQUEST_NULL;
      pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                           &peUpdates);
      MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
                UPDATE_TAG, MPI_COMM_WORLD,
      pendingUpdates -= peUpdates;
    }
  }
}
/* send remaining updates in buckets */
while (pendingUpdates > 0) {
  /* receive messages */
  do {
    MPI_Test(&inreq, &have_done, &status);
    if (have_done) {
      if (status.MPI_TAG == UPDATE_TAG) {
        MPI_Get_count(&status, tparams.dtype
        bufferBase = 0;
        for (j=0; j < recvUpdates; j ++) {
          inmsg = LocalRecvBuffer[bufferBase
          LocalOffset = (inmsg & (tparams.T
                         tparams.GlobalStart
          HPCC_Table[LocalOffset] ^= inmsg;
        }
      } else if (status.MPI_TAG == FINISHED
        /* we got a done message. Thanks for playing. */
        NumberReceiving--;
      } else {
        MPI_Abort( MPI_COMM_WORLD, -1 );
      }
      MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
    }
  } while (have_done && NumberReceiving > 0);
MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
if (have_done) {
  outreq = MPI_REQUEST_NULL;
  pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                       &peUpdates);
  MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
            UPDATE_TAG, MPI_COMM_WORLD, &outreq);
  pendingUpdates -= peUpdates;
}
}
}
MPI_Abort( MPI_COMM_WORLD, -1 );
}
MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
          MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
}
MPI_Waitall( tparams.NumProcs, tparams.finish_req, tparams.finish_statuses);
```

```
…
forall (_, r) in zip(Updates, RAStream()) do
  T[r & indexMask].xor(r);
…
```

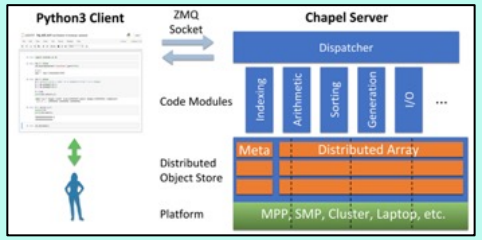


Applications of Chapel

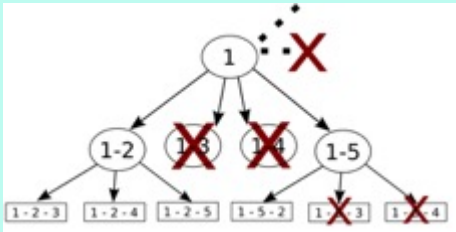
Active GPU efforts



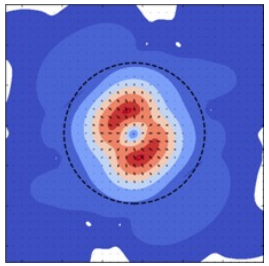
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



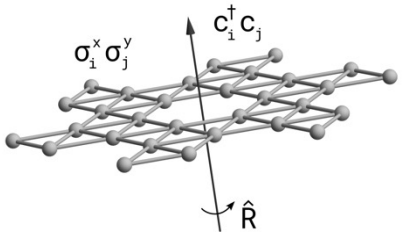
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



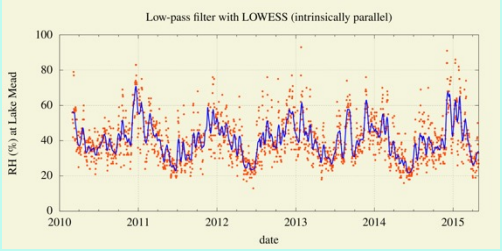
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

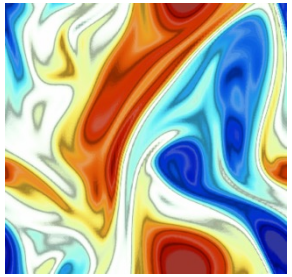


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

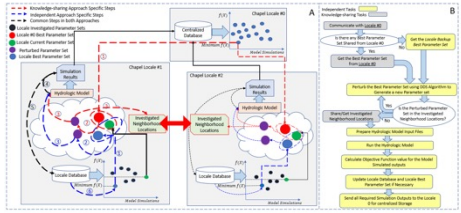


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

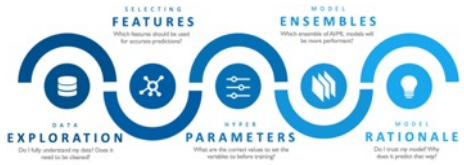
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



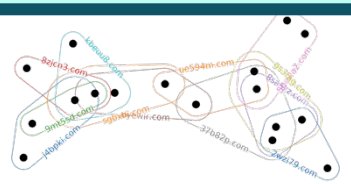
ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.



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

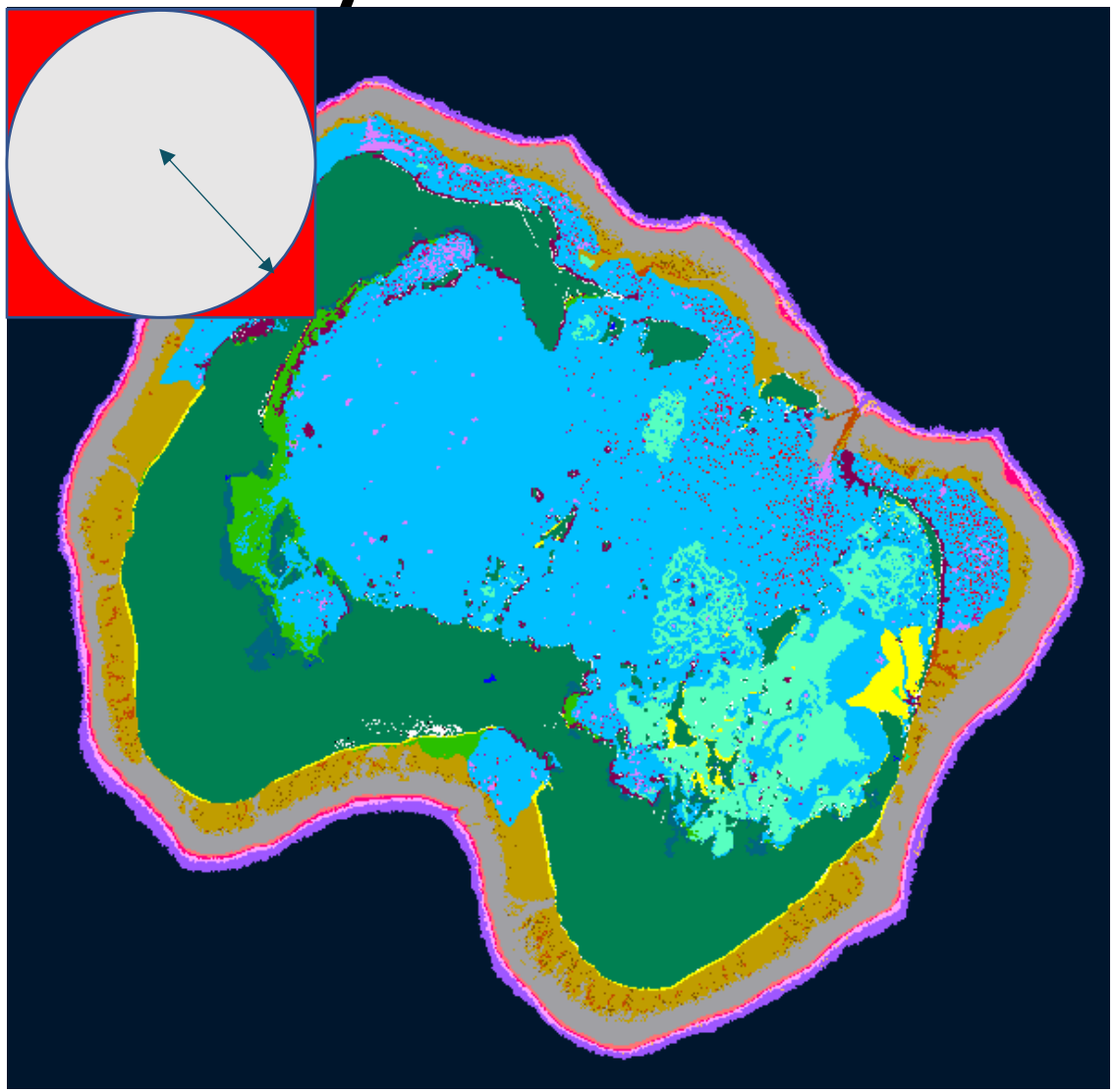
?

Your Application Here?

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

Use Case: Image Processing for Coral Reef Biodiversity

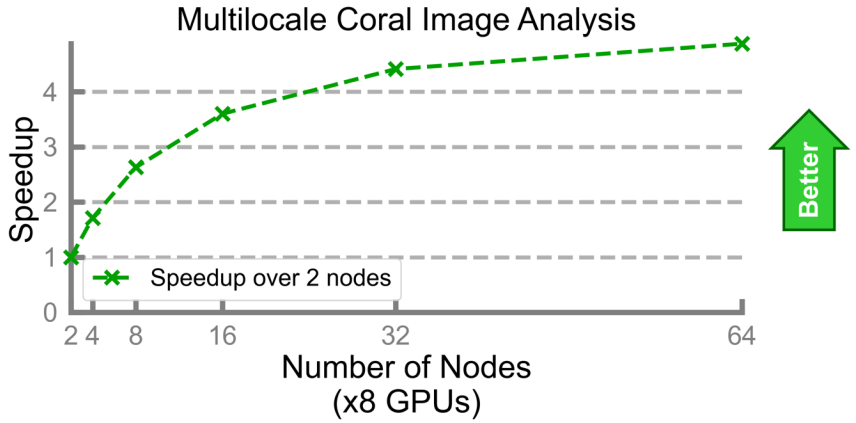
- **Analyzing images for coral reef biodiversity**
  - Important for prioritizing interventions
- **Algorithm implemented productively**
  - Add up weighted values of all points in a neighborhood, i.e., convolution over image
  - Developed by Scott Bachman, NCAR scientist who is a visiting scholar on 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



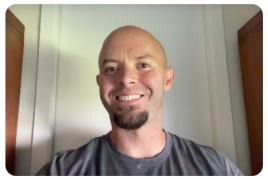
Use Case: Image Processing for Coral Reef Biodiversity

Runs on Frontier!

- At 64 nodes, takes 20 minutes
  - As opposed to ~27 days on a laptop
- Straightforward code changes:
  - from sequential Chapel code
  - to GPU-enabled one
  - to multi-node, multi-GPU, multi-thread



Read a recent interview with Scott Bachman on Chapel Blog



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

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 Scott started as a visiting scholar with the Chapel team at HPE, and it is just one of several projects he took on during his time with us. Since wrapping up his visit at HPE, Scott has continued to apply Chapel in his work, which he describes below.

One noteworthy thing about the computation Scott describes here is that it is just a few hundred lines of Chapel code, yet can be used to drive the CPUs and GPUs of the world's largest supercomputers. This serves as a sharp contrast with the 100+k lines that make up the CHAMPS framework covered in our previous interview. Together, the two demonstrate the vast spectrum of code sizes that researchers are productively writing in Chapel.

What's In Store Today

Coming up Next: A live demo

- Introduction to the language and parallelism
- Showcase of GPU capabilities
- Inference using ChAI (Chapel AI Library)

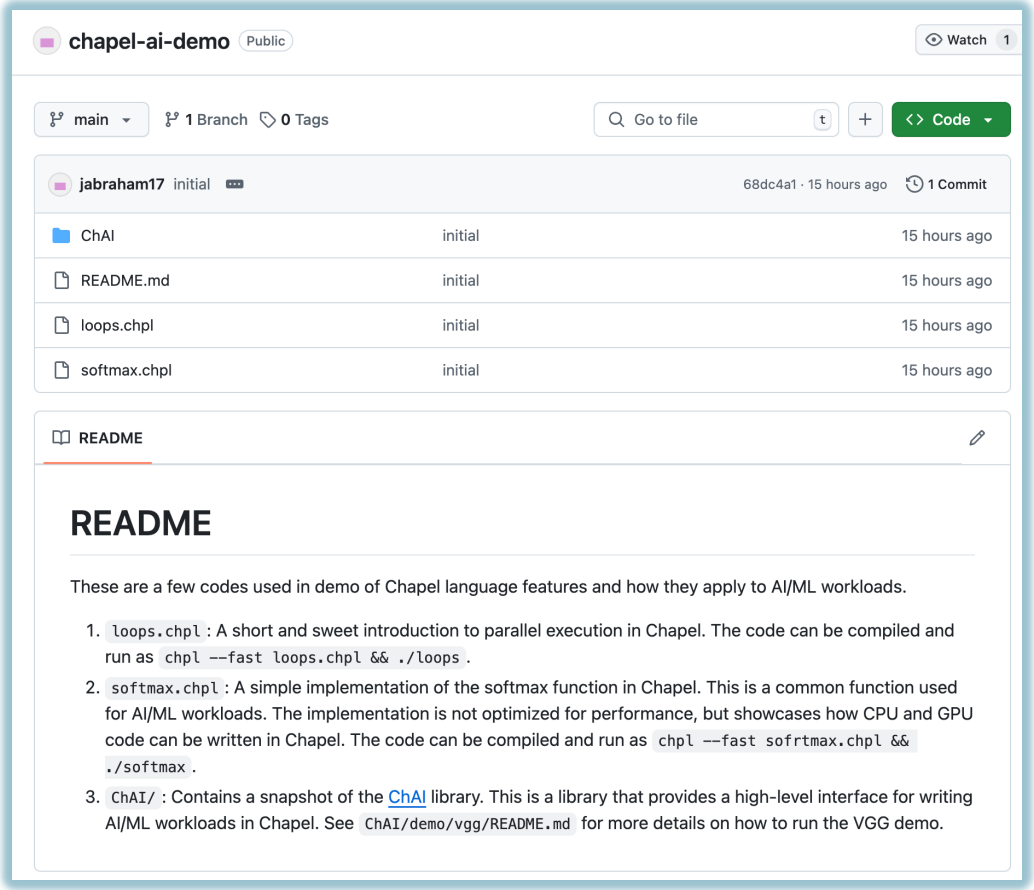
Later on: Ways to contribute to the Chapel Universe

- A spectrum of contribution opportunities, technical and social alike

Live Demo

Example Codes Are Available

https://github.com/jabraham17/chapel-ai-demo/



Contributing to The Chapel Universe

Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

SOCIAL

Join the discussion

Follow us on social media

Help expand the Chapel universe!

Contributing to The Chapel Universe

TECHNICAL

SOCIAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

Join the discussion

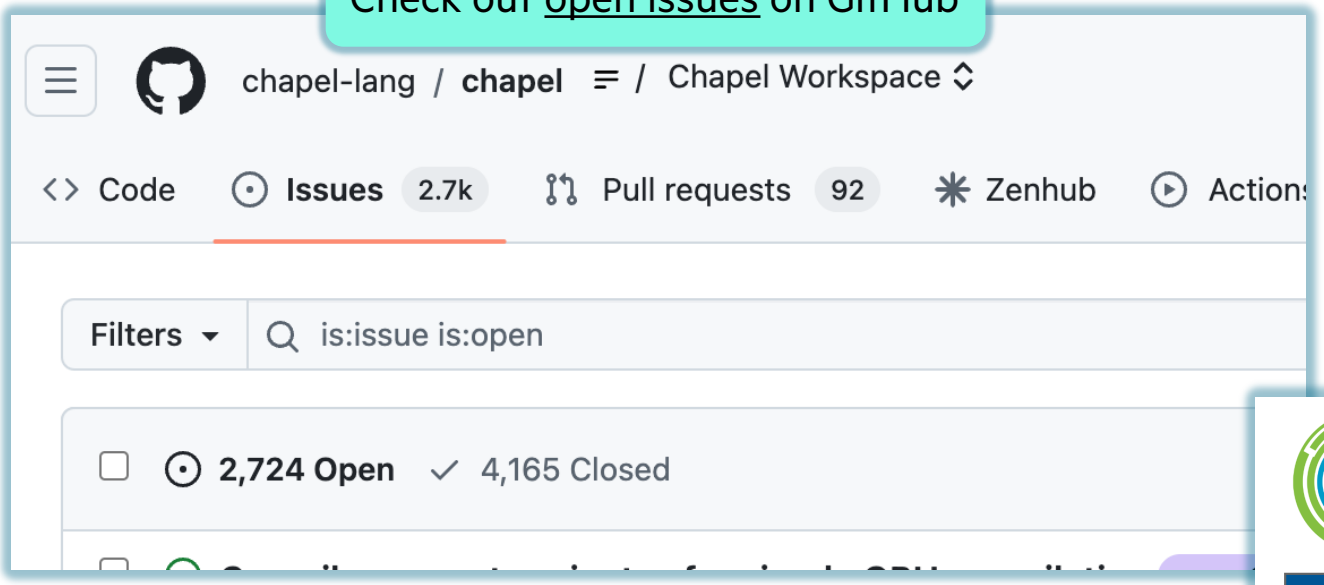
Follow us on social media

Help expand the Chapel universe!

Contribute to the Standard Modules, Runtime or Compiler

Chapel has had more than 200 contributors so far!

Check out open issues on GitHub

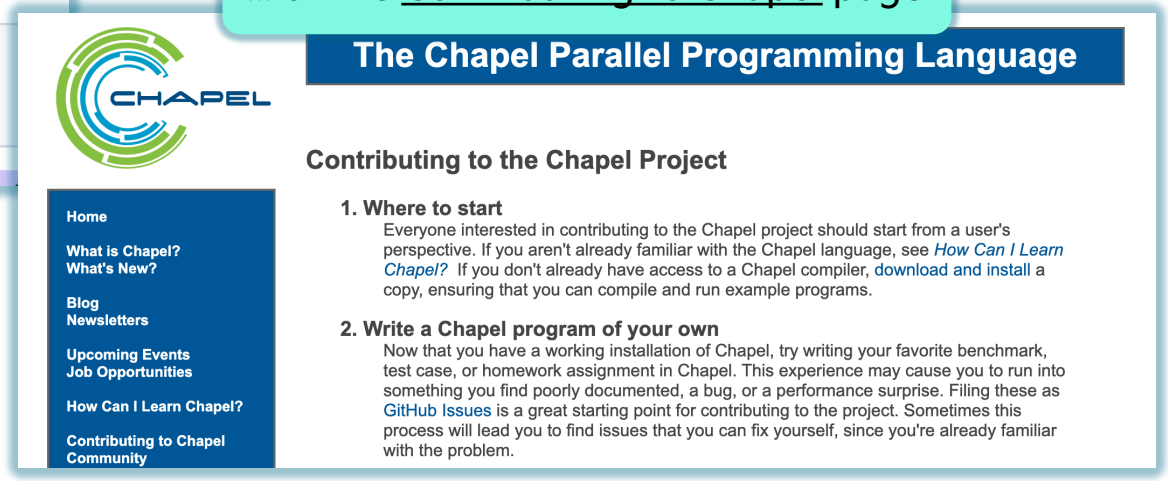


Tip: Use labels to filter open issues; such as

easy / straightforward

good first issue

... or the Contributing to Chapel page



Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

Join the discussion

SOCIAL

Follow us on social media

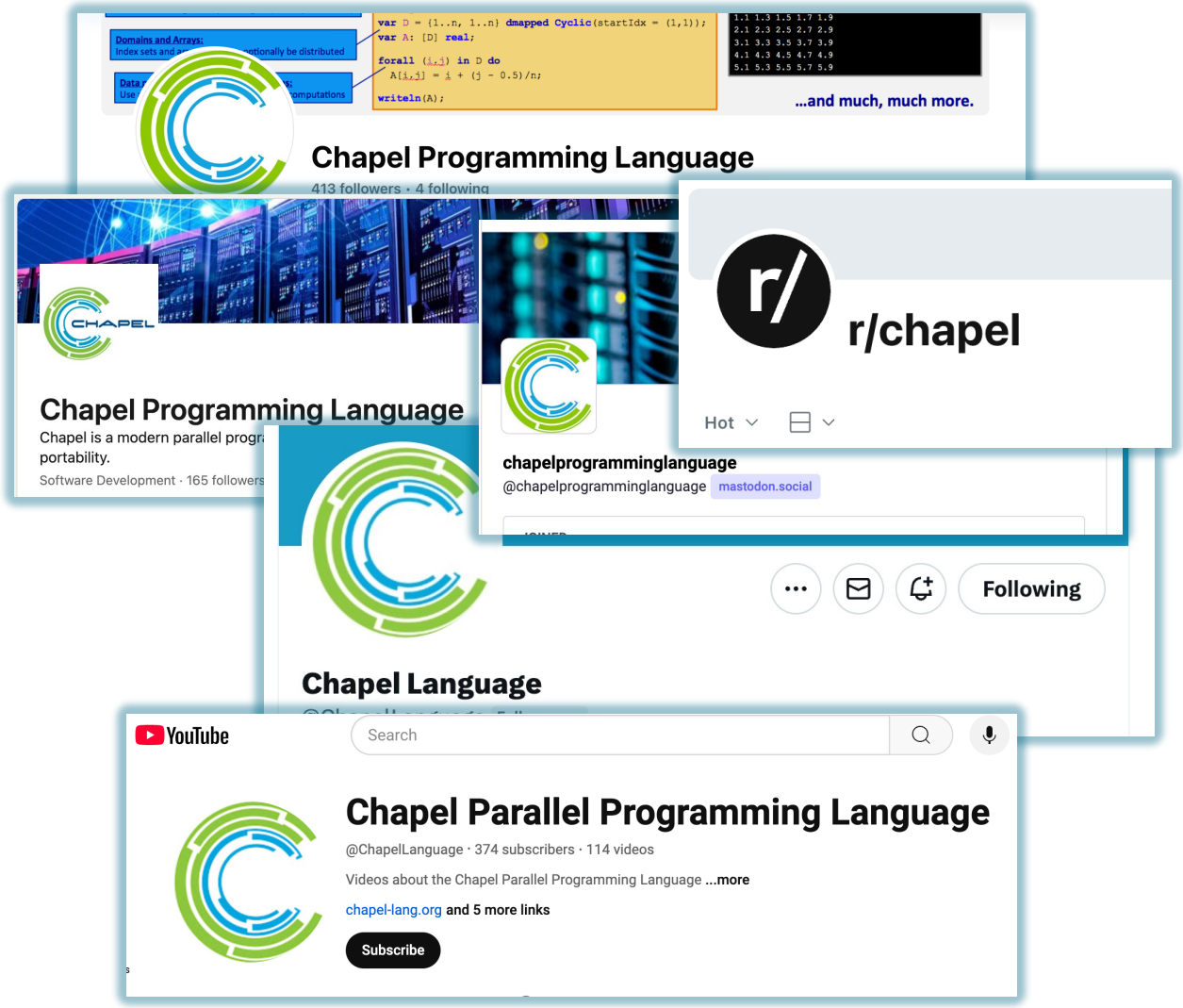
Help expand the Chapel universe!

Follow Chapel on Social Media

We have accounts on the following platforms:

- LinkedIn: ChapelLanguage
- Mastodon: @ChapelProgrammingLanguage
- X / Twitter: @ChapelLanguage
- Facebook: @ChapelLanguage
- YouTube: @ChapelLanguage
- Reddit: r/Chapel

- There is weekly activity on these accounts:
  - Upcoming Chapel events, talks, papers from the community
  - New Chapel resources, tutorials, and demos
  - Updates from releases, performance studies
- Look out for Chapel news on other platforms
  - e.g. Hacker News and Lobsters
- Follow, like, repost, **amplify the message!**



Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

SOCIAL

Join the discussion

Follow us on social media

Help expand the Chapel universe!

Chapel Makes HPC and Parallel Programming More Accessible

Chapel **removes the layers of complexity** for code writing, allowing seamless code development **while maintaining computational performance**. It is **ideal to develop very large and complex computational models**.

Éric Laurendeau, Professor of Mechanical Engineering, Polytechnique Montreal

With the coral reef program, I was able to **speed it up by a factor of, like 10,000**. I would say some of that was algorithmic… but again, Chapel had the **features in the language that allowed me to do it pretty succinctly**.

Scott Bachman, Oceanographer, [C]Worthy

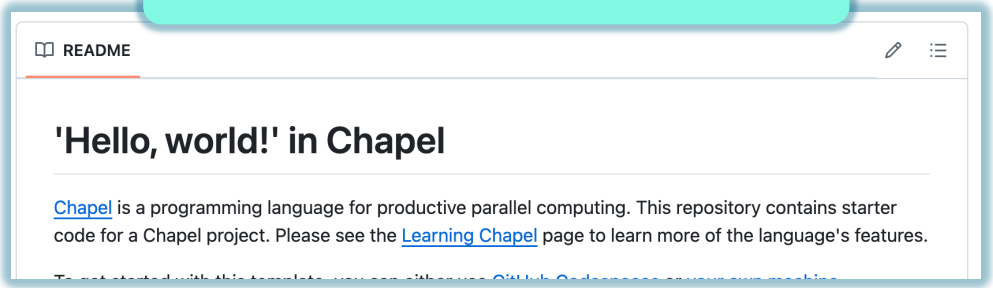
A lot of the nitty gritty is hidden from you until you need to know it. … It feels like **the complexity grows as you get more comfortable** -- rather than being hit with everything at once.

Tess Hayes, Developer, Bytoa

Write Your Next Application in Chapel!

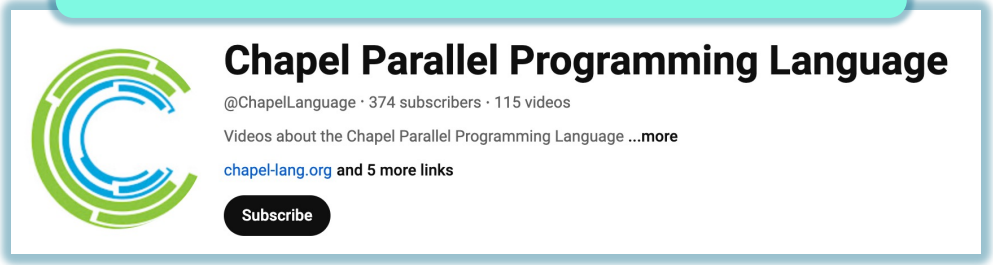
- Do you have applications that can benefit from a **parallel-first language** that
  - has intuitive, baked-in parallelism and locality features
  - can be used on laptops, supercomputers or anything in between
  - can run on NVIDIA and AMD GPUs in a vendor-neutral way
- Maybe that application is;
  - implemented in Python, MATLAB, R, and
    - can benefit from parallelism
    - without complicating the code
  - implemented in C/C++/Fortran + MPI + XYZ, and is getting
    - harder to maintain,
    - harder to onboard new developers
  - not implemented, yet!

Try Chapel on GitHub Codespaces

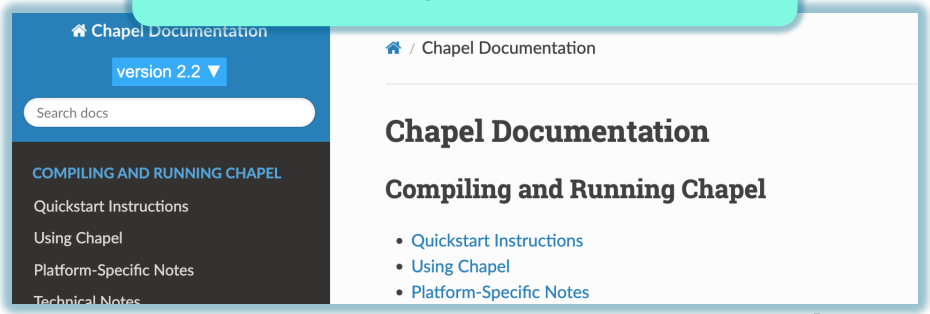


See many other ways of trying Chapel
chapel-lang.org/download.html

Check out tutorials and demos on YouTube



Check out Chapel Documentation



Get Familiar with Open-Source Projects using Chapel

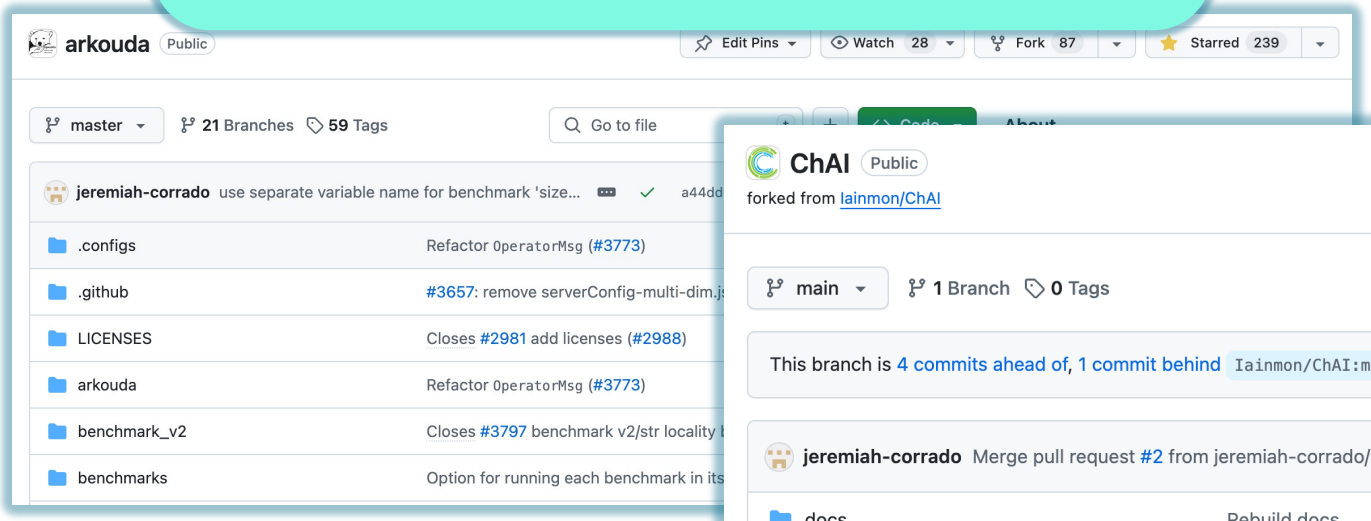
Browse through projects with "chapel" tag on GitHub: github.com/topics/chapel

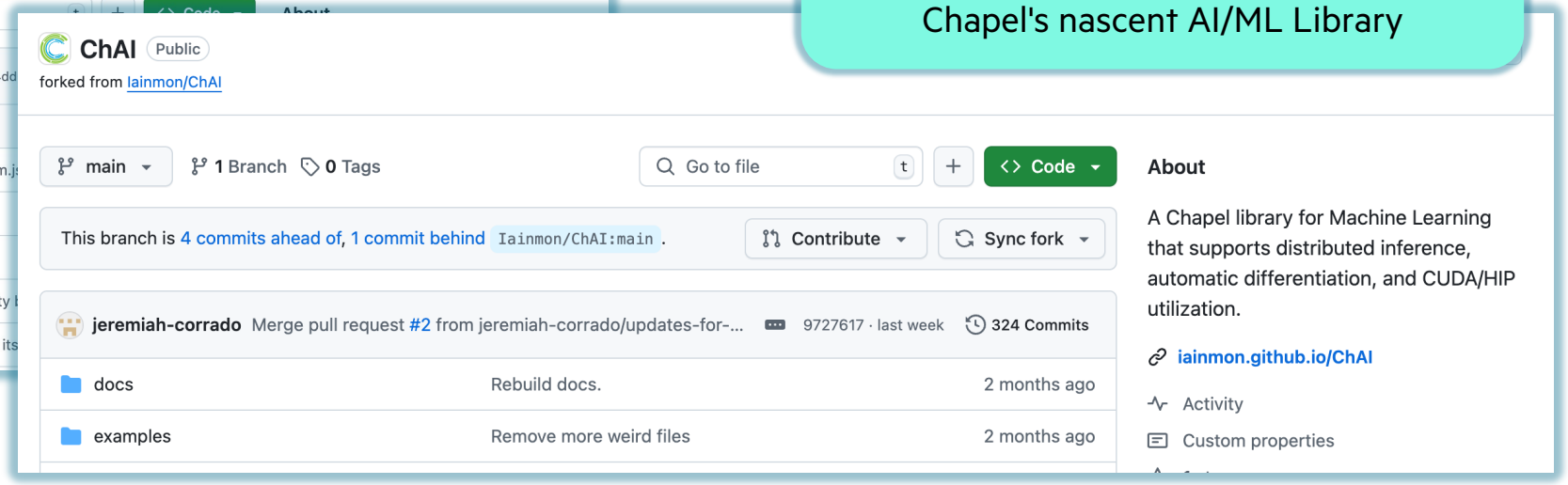
Arkouda: Interactive Data Analytics at Supercomputing Scale

A numpy/pandas-like Python library backed by a server implemented in Chapel

ChAI: Chapel AI Library

Chapel's nascent AI/ML Library





Create Your Favorite Library in Chapel

Check out this wishlist of libraries for inspiration: github.com/chapel-lang/chapel/issues/6329

- … or any other library from any other language

Chapel's package manager, Mason, can help

Mason

Overview

This guide on Mason Packages contains everything that you need to know to hit the ground developing in Chapel using mason, whether that be to create a library to contribute to the community, use an existing package, or just to help with the process of building your Chapel programs.

For example, see this automatic differentation library

README MIT license

ForwardModeAD

license MIT docs latest lifecycle maturing

Lightweight library for forward-mode automatic differentiation using dual numbers and functions overloading. It can c
as a combination of

… or this Interval Arithmetic one

README MIT license

CHIMERA: CHapel Interval MEthods libRAry

A library for interval arithmetic and applications in Chapel.

Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

SOCIAL

Join the discussion

Follow us on social media

Help expand the Chapel universe!

Contribute to Chapel Tooling

Consider helping us improve Chapel's tools, such as:

Chapel Language Server (CLS)

Chapel Linter (chplcheck)

VSCode Extension

Python Bindings for the Chapel Compiler (chapel-py)

Mason: Chapel Package Manager (written in Chapel!)

... or consider other editor extensions, tools that you find useful.

Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

Join the discussion

SOCIAL

Follow us on social media

Help expand the Chapel universe!

Join our Community Channels and Discussions

For more technical discussion and/or support:

- Discourse: https://chapel.discourse.group/
- Gitter: https://gitter.im/chapel-lang/chapel
- Stack Overflow: https://stackoverflow.com/questions/tagged/chapel
- GitHub Issues: https://github.com/chapel-lang/chapel/issues
- Monthly Office Hours / Live Demos: https://chapel-lang.org/events.html

Upcoming Chapel-related Events

Also check out the Chapel Community Calendar (ICS), which is a published Outlook calendar with most of the events below.

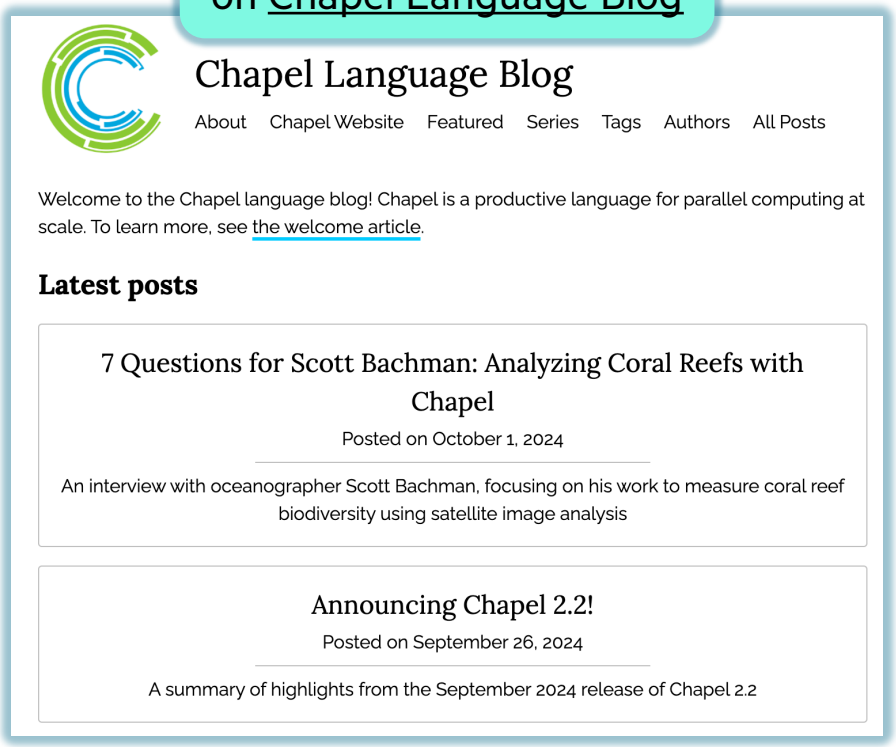
2024

- **October 3:** (10–11 AM PT): **Chapel Demo Session**
  Topic: Chapel's new 'Image' module
  [See the Community Calendar above to join the Teams meeting]
- **October 17** (10–11 AM PT): **Chapel Office Hours**
  [See the Community Calendar above to join the Teams meeting]
- **December 19**: Anticipated release date of Chapel 2.3

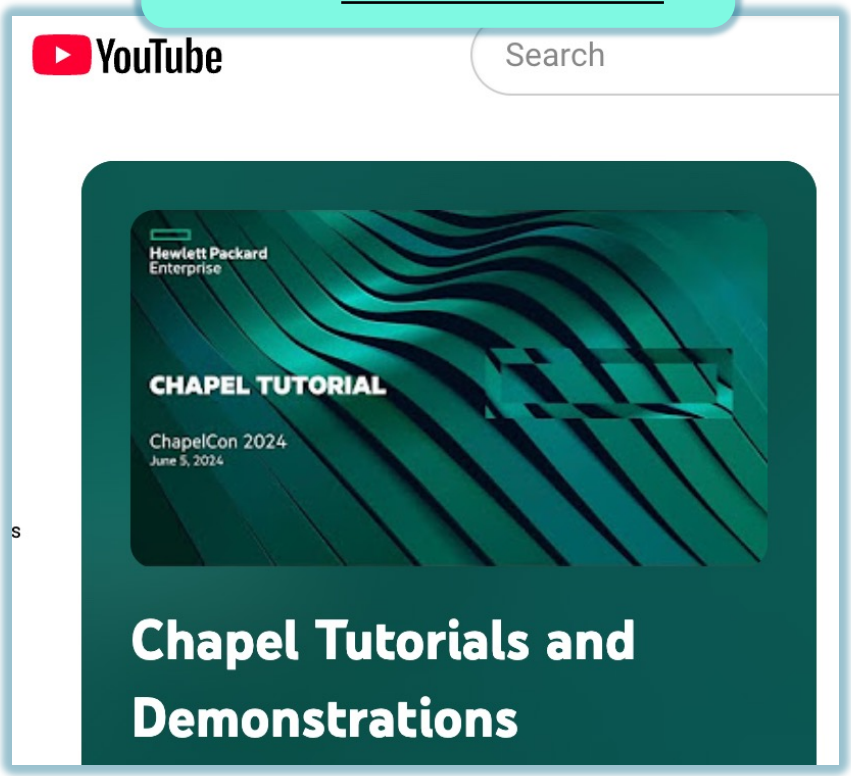
Let Us Know About Your Work Using Chapel

We would love to boost your work…

By publishing your article on Chapel Language Blog



… or by posting your demo on our YouTube channel



… and other social media platforms.

Contributing to The Chapel Universe

TECHNICAL

Contribute to the core language

Contribute to tooling

Write your favorite application or library in Chapel

SOCIAL

Join the discussion

Follow us on social media

Help expand the Chapel universe!

Help expand the Chapel universe!

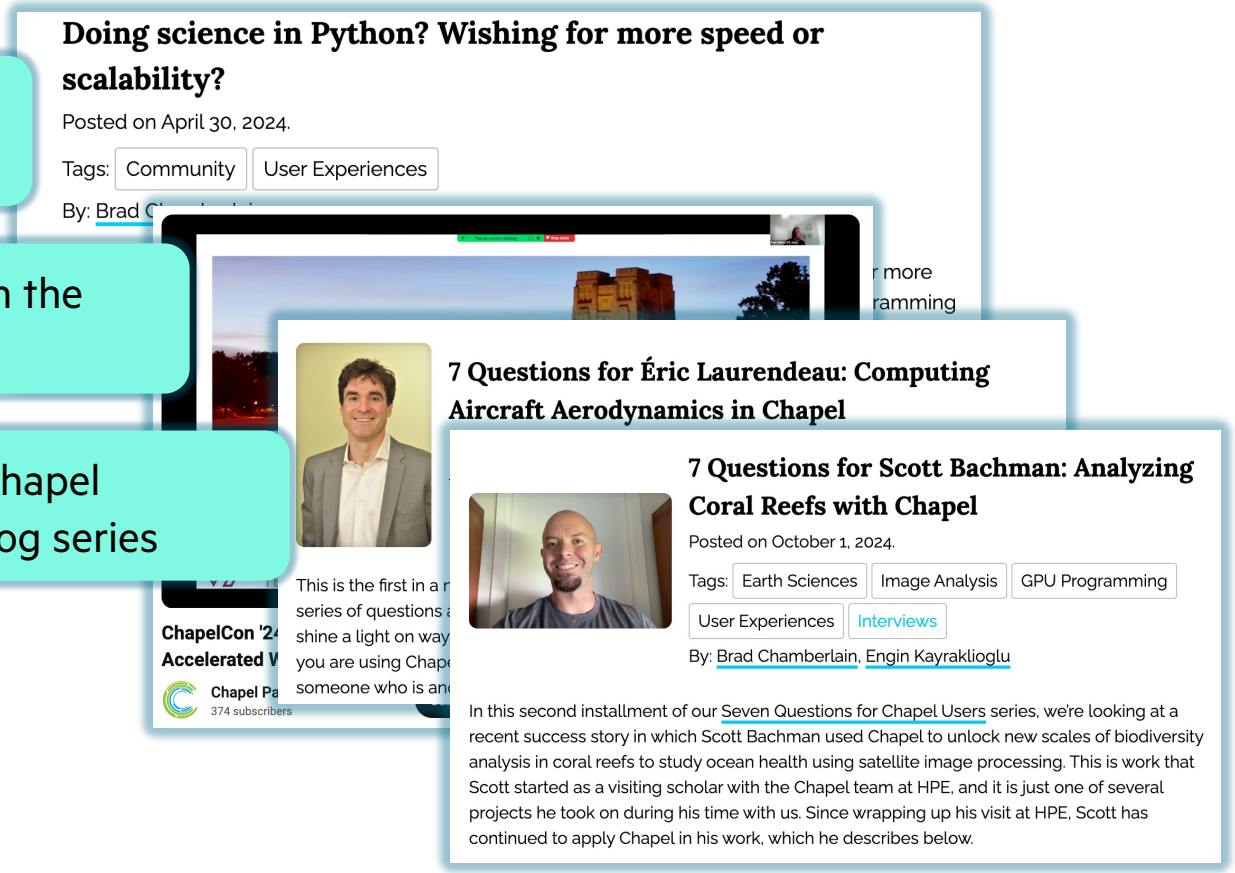
- Chapel language is used in **industry**, **academia** and **government**
- The current focus of the Chapel team @HPE is to expand our community further

Check out the following for further inspiration…

This open call by Chapel's Tech Lead Brad Chamberlain

The ChapelCon '24 Keynote by Paul Sathre on the importance of Parallel-First Languages

What our users have to say about Chapel on the 7 Questions for Chapel Users blog series



Thanks!