Vendor-Neutral, Scalable GPU Programming with Chapel

Engin Kayraklioglu, HPE

HPSFCon 2026
March 16, 2026

What is Chapel?

Chapel is a programming language that is:

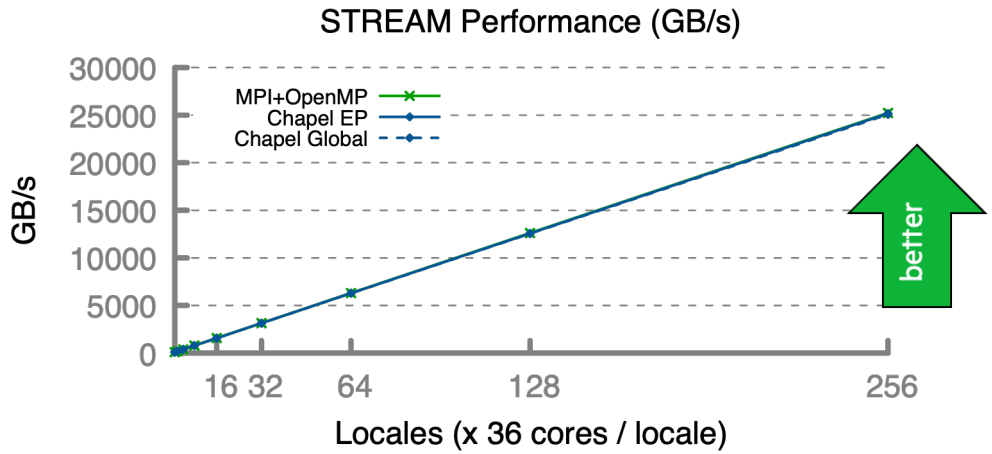
- **parallel & scalable:**
  - users benefit from multicore parallelism, GPUs, supercomputers…
- **portable:**
  - runs on Linux, MacOS, WSL, Cloud, InfiniBand, Slingshot, Raspberry Pi…
- **general-purpose:**
  - used in fields like aerodynamics, data and graph analytics, various earth sciences…
- **modern:**
  - object-orientation, memory and type safety, error handling, tool/IDE support…
- **open-source:**
  - a member of High-Performance Software Foundation & Linux Foundation…

HPCC Benchmarks in C+MPI+OpenMP vs. Chapel

STREAM TRIAD: C + MPI + OPENMP

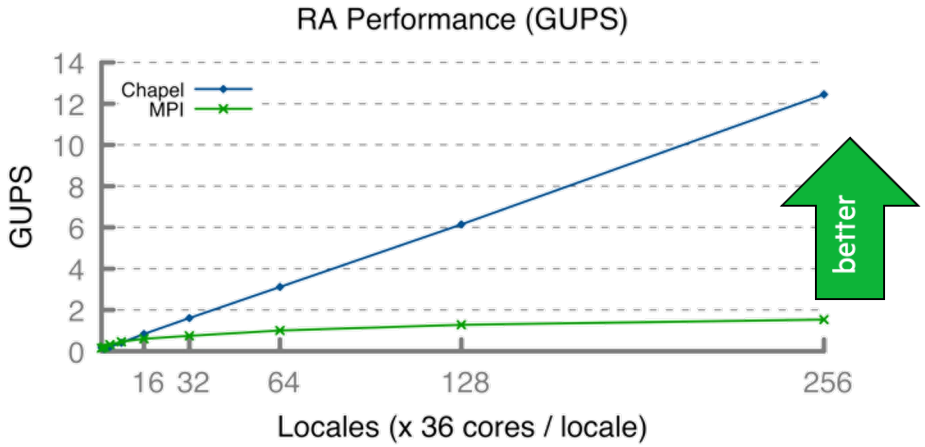
```
#include <hpcc.h>
#ifdef _OPENMP
#include <omp.h>
#endif

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;
```



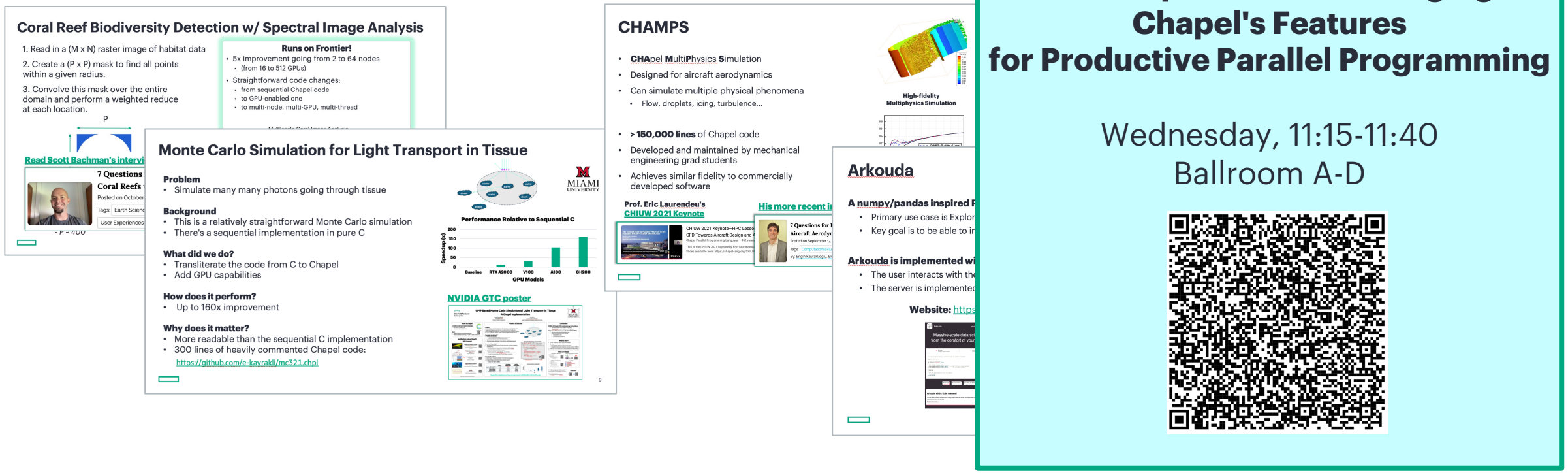
HPCC RA: MPI KERNEL

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



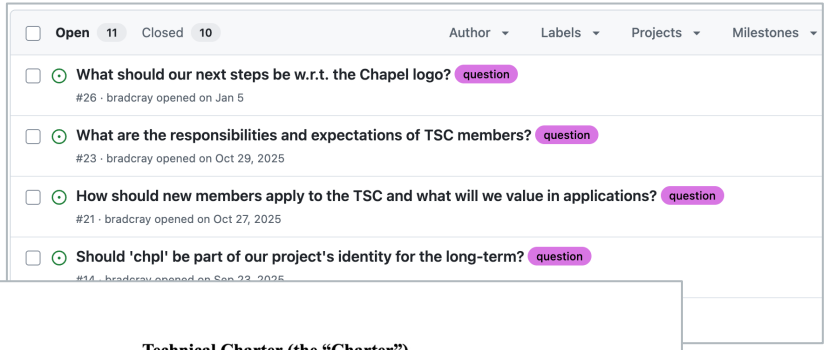
Chapel Applications

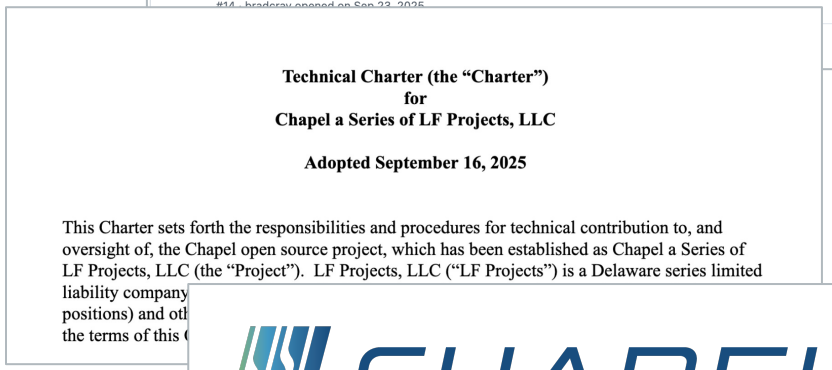
- Applications written in Chapel:
  - range from couple of hundreds of lines of code to >150k lines
  - run on workstations, single GPUs, commodity clusters, leadership-class supercomputers
  - help scientists, engineers, data analysts



Chapel & HPSF

- In the ~last year as part of our efforts to become an HPSF member we:
  - Formed a Technical Steering Committee
    - As part of that, we created a public repo for notes & discussions
  - Completed the Technical Charter and updated GOVERNANCE docs
  - Became an Established HPSF project ☺
  - Changed our logo
  - Opened weekly project meetings and deep-dives to the public



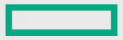


Technical Charter (the "Charter")
for
Chapel a Series of LF Projects, LLC

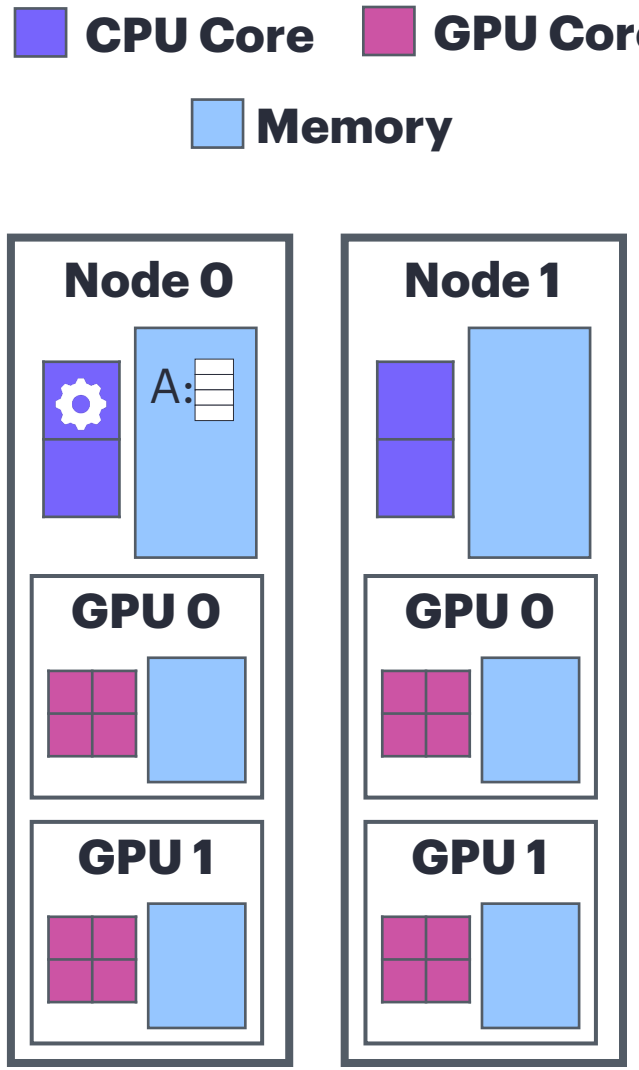
Adopted September 16, 2025

This Charter sets forth the responsibilities and procedures for technical contribution to, and oversight of, the Chapel open source project, which has been established as Chapel a Series of LF Projects, LLC (the "Project"). LF Projects, LLC ("LF Projects") is a Delaware series limited liability company
positions) and oth
the terms of this

What does Chapel code look like?



Chapel Code: Fundamentals



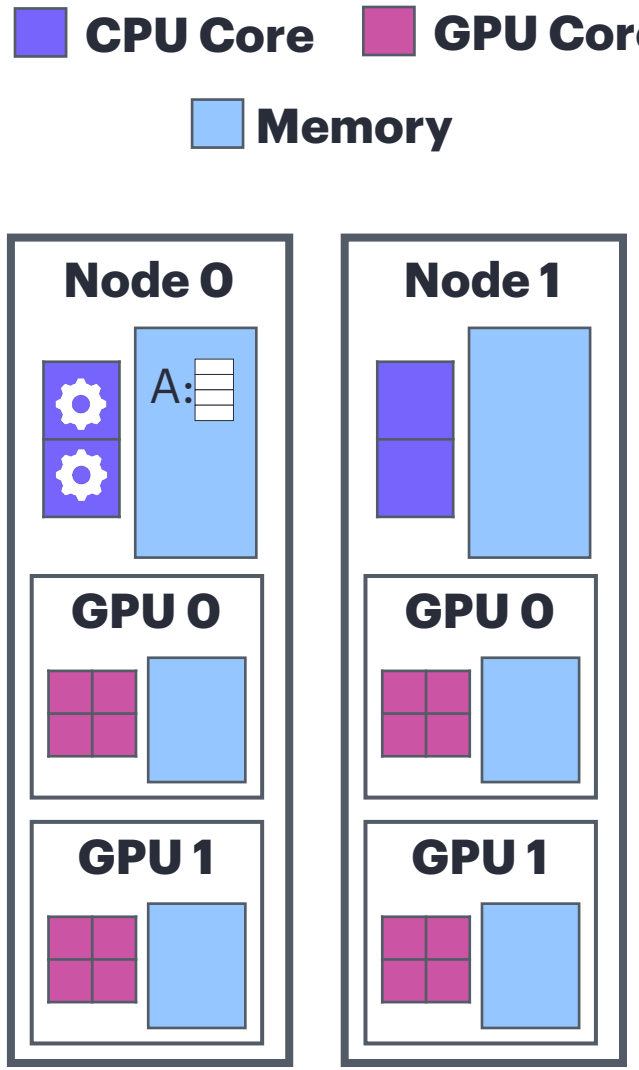

```
var A: [1..10] int;
```

Local, non-distributed array allocation

```
for elem in A do
  elem += 1;
```

Sequential iteration over the array

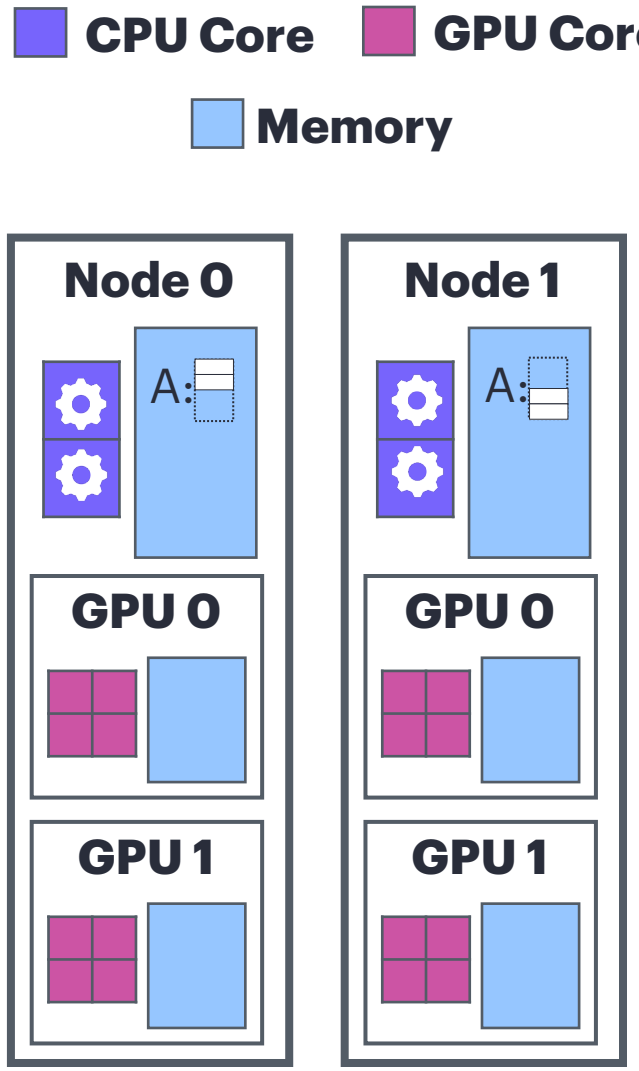
Chapel Code: Basic Data Parallelism




```
var A: [1..10] int;

forall elem in A do
  elem += 1;
```

Parallel iteration over the array

Chapel Code: Basic Data Parallelism



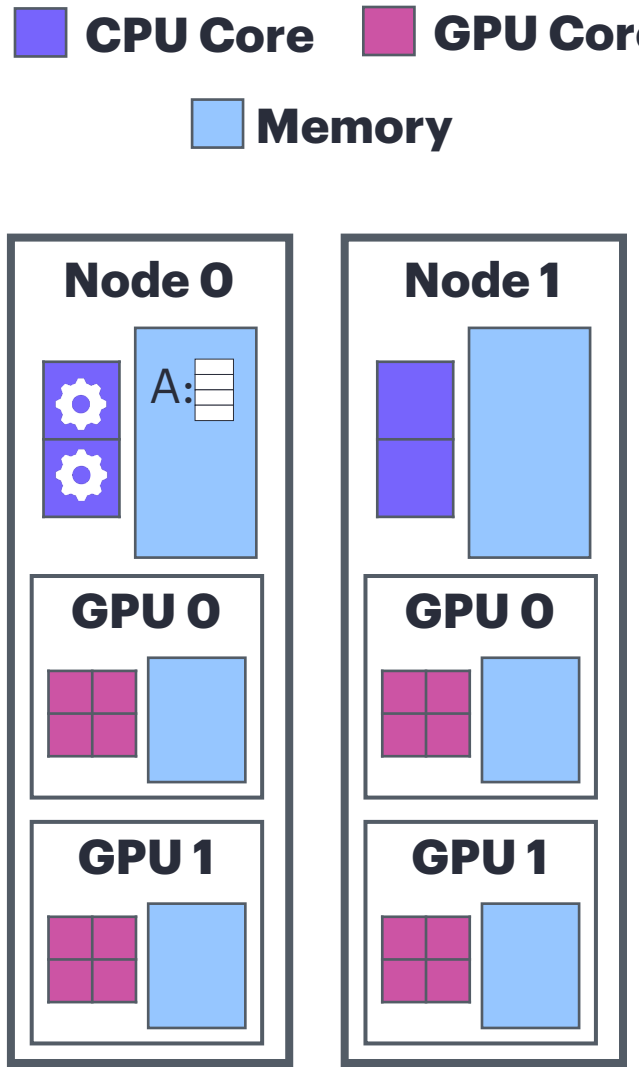

```
use BlockDist;
var Arr = blockDist.createArray(1..10, int);
```

Block-distributed array allocation

```
forall elem in Arr do
  elem += 1;
```

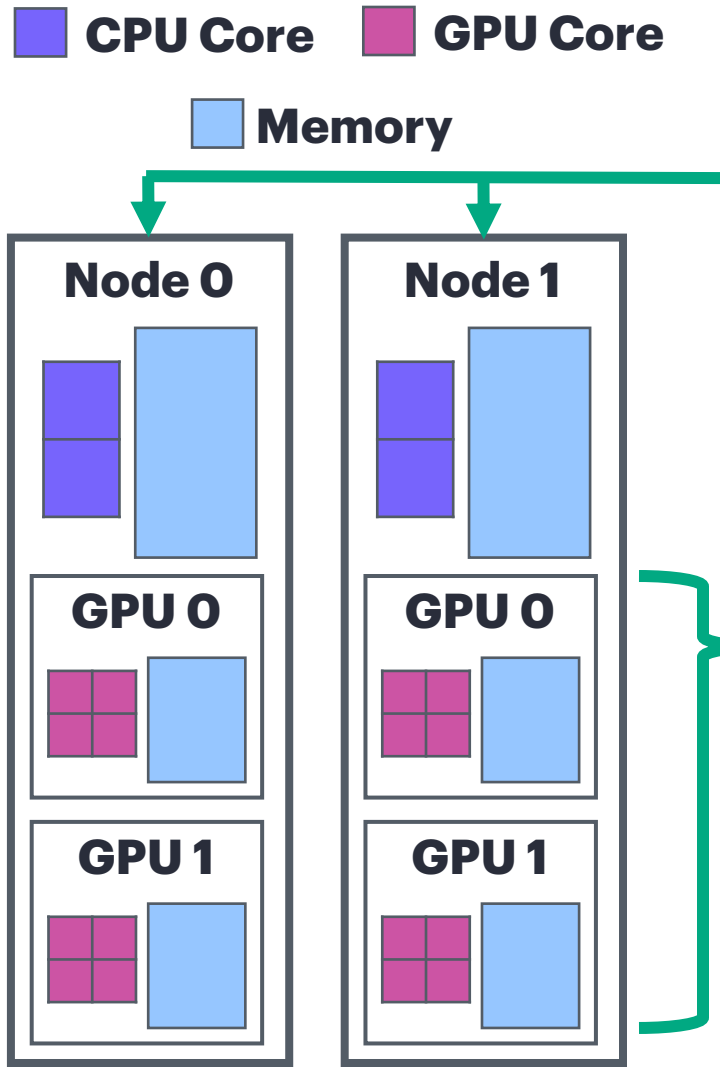
Distributed, parallel iteration over the array

Chapel Code: Basic Data Parallelism




```
var A: [1..10] int;

forall elem in A do
  elem += 1;
```

Chapel Code: Locality as a First-Class Citizen



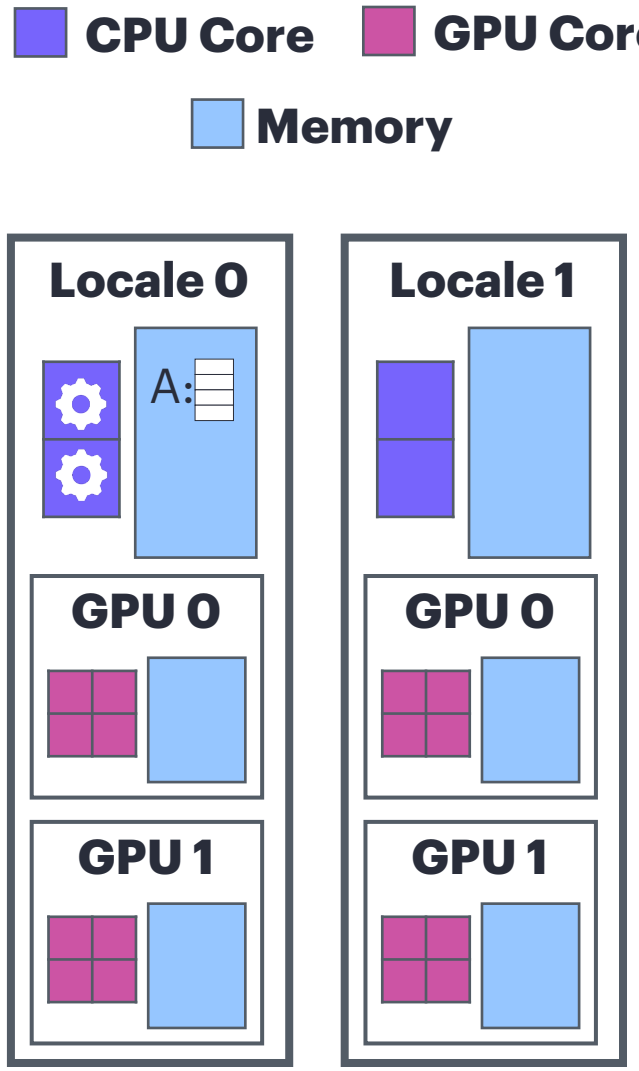
- "locale" is a built-in Chapel type
- In the most common scenario: locale == node
- Under the hood, a locale is always a process
- Users can configure their applications to use:
  - a locale per socket, GPU, NUMA domain...

- GPUs are represented as "sublocales"

Predefined Variables for Locality

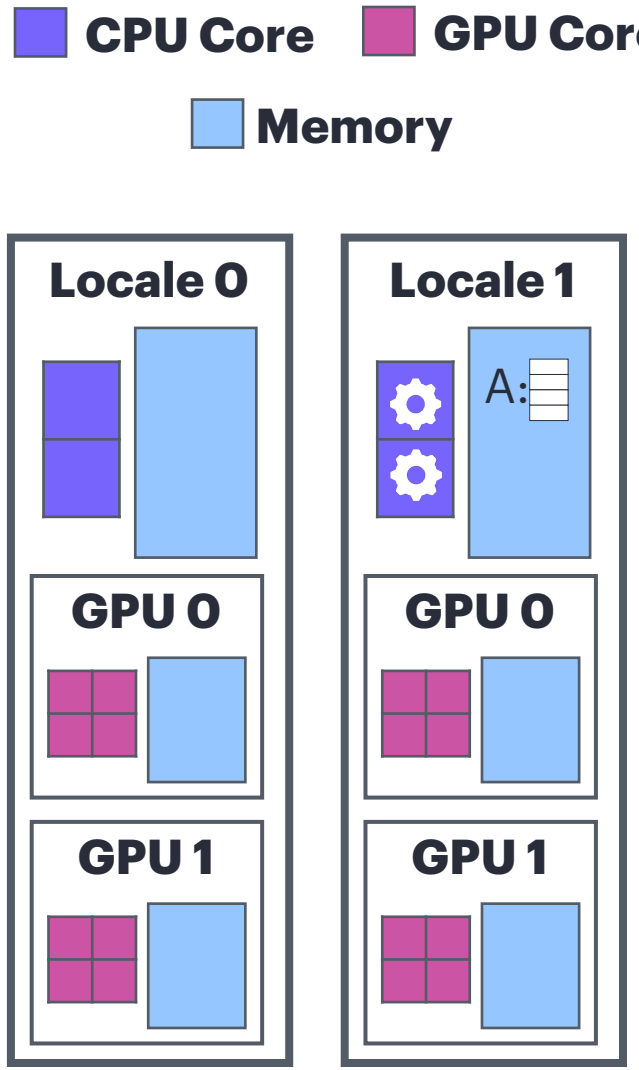
`here`	The current execution locale
`Locales`	An array storing all locales
`locale.gpus`	An array storing all GPU sublocales

Chapel Code: Basic Data Parallelism + Locality




```
var A: [1..10] int;

forall elem in A do
  elem += 1;
```

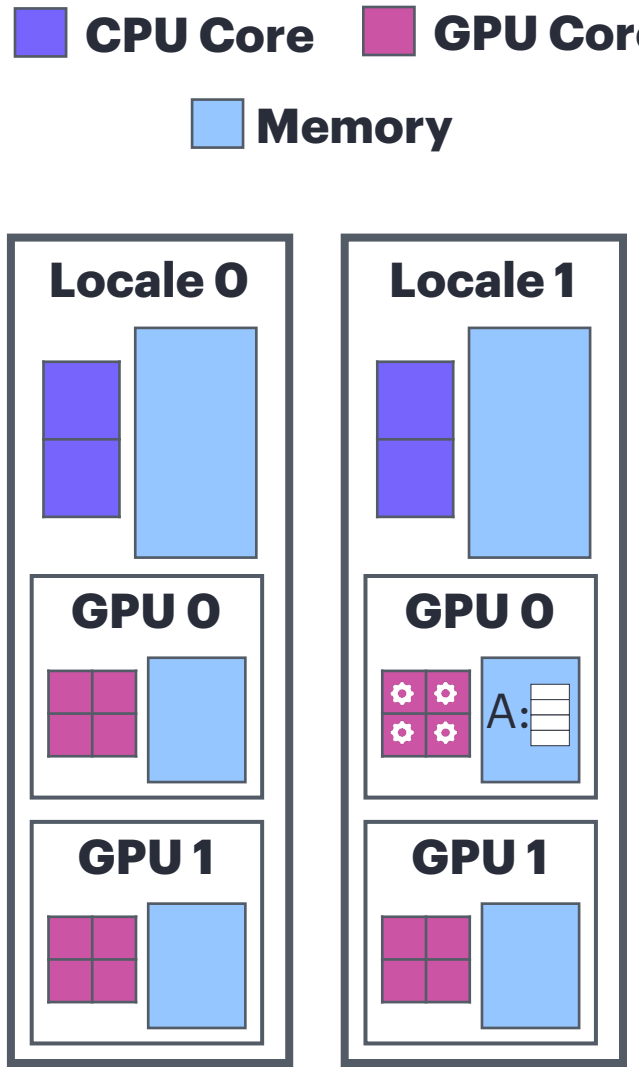
Chapel Code: Basic Data Parallelism + Locality




```
on Locales[1] {
  var A: [1..10] int;

  forall elem in A do
    elem += 1;
}
```

The 'on' statement
moves the execution to a remote locale

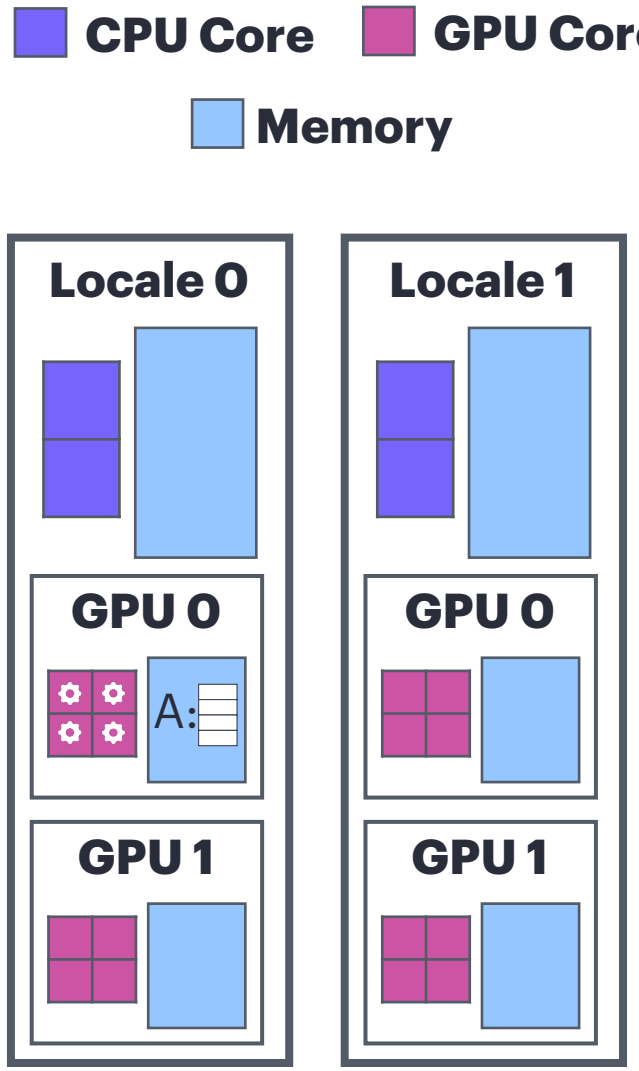
Chapel Code: Basic Data Parallelism + Locality




```
on Locales[1].gpus[0] {
  var A: [1..10] int;

  forall elem in A do
    elem += 1;
}
```

Each locale object has a
'gpus' array
which stores GPU sublocales

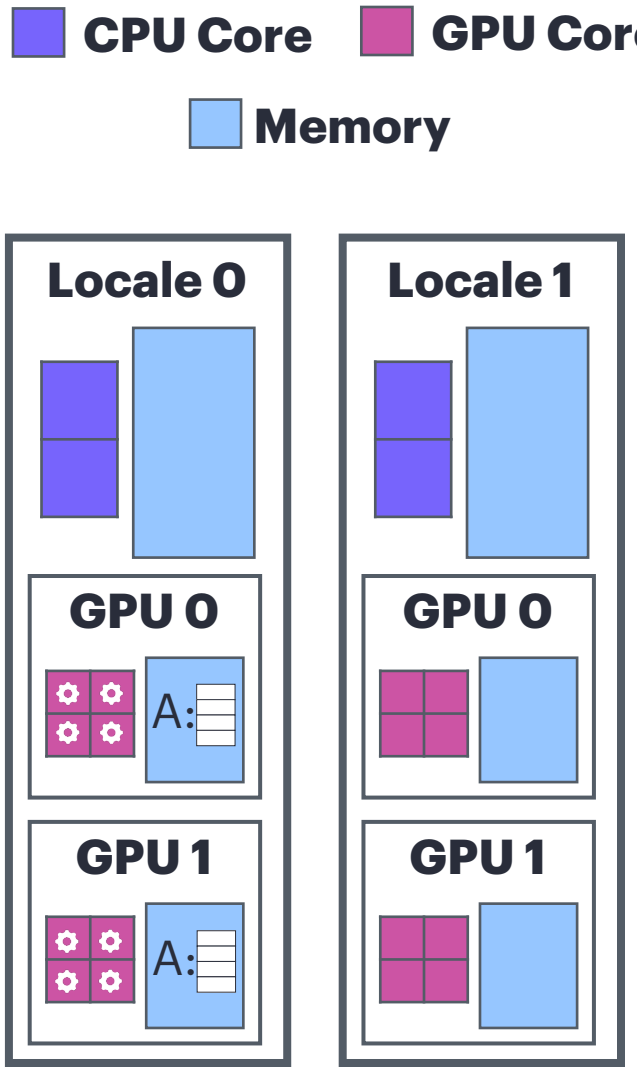
Chapel Code: Basic Data Parallelism + Locality



'here' is a built-in representing the current execution locale

```
on here.gpus[0] {
  var A: [1..10] int;

  forall elem in A do
    elem += 1;
}
```

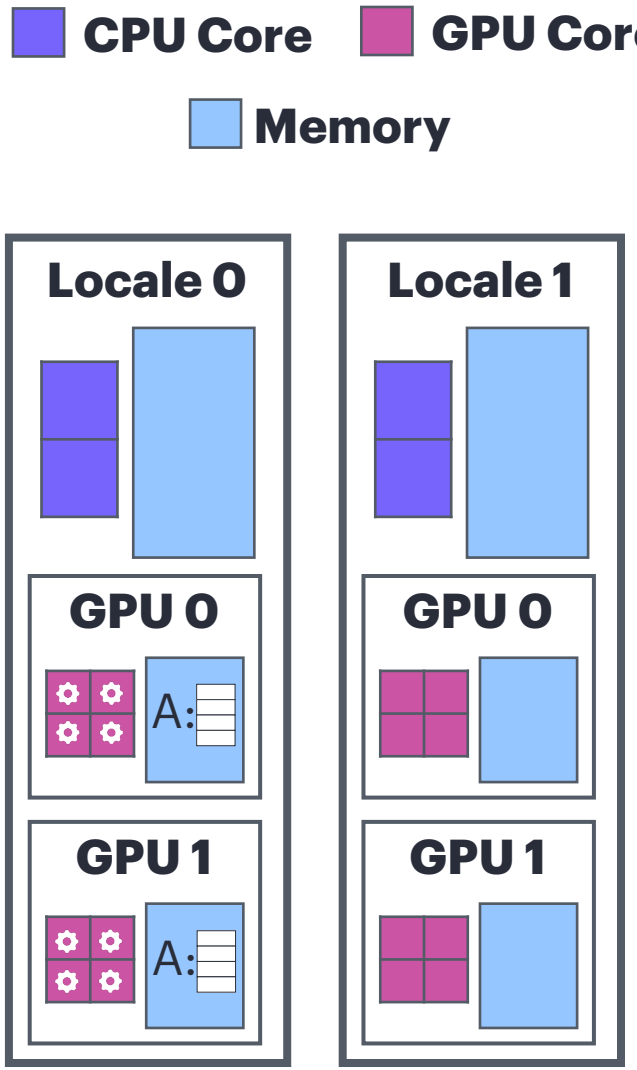
Chapel Code: Data + Task Par. + Locality



'coforall' loops run each iteration in a parallel task

```
coforall gpu in here.gpus {
  on gpu {
    var A: [1..10] int;

    forall elem in A do
      elem += 1;
  }
}
```

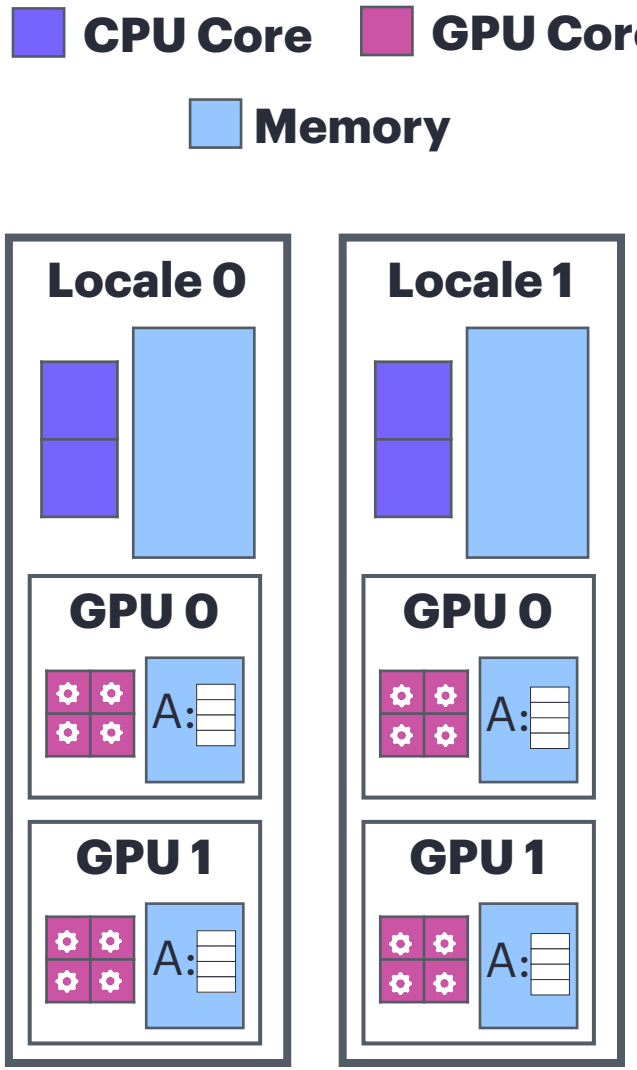
Chapel Code: Data + Task Par. + Locality



'do' can be used instead of curly braces for single-statement blocks
(this is just a matter of style)

```
coforall gpu in here.gpus do on gpu {
  var A: [1..10] int;

  forall elem in A do
    elem += 1;
}
```

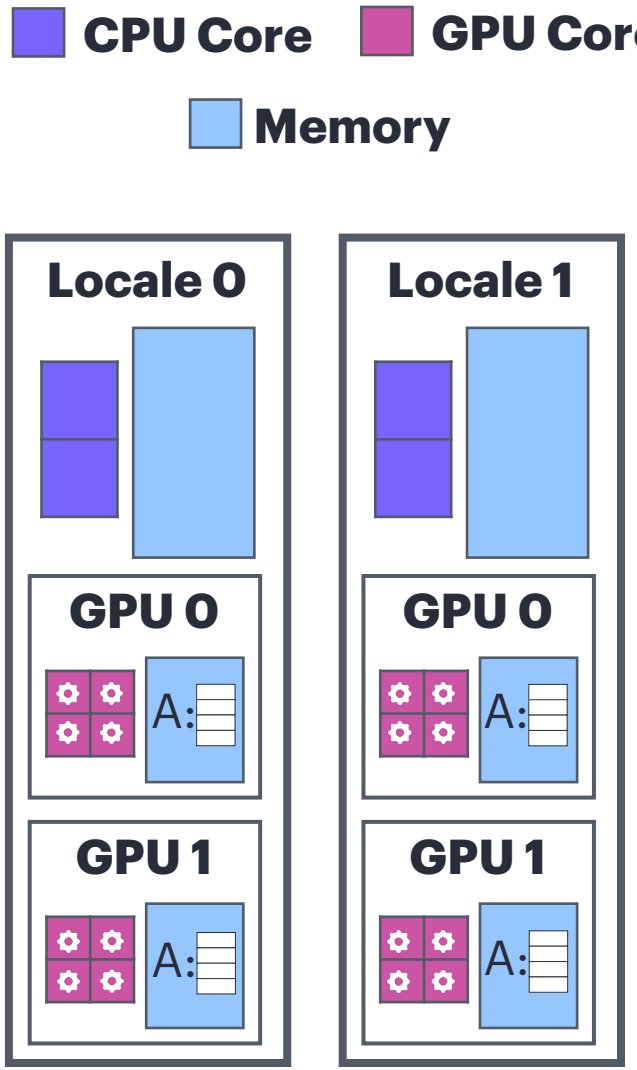
Chapel Code: Data + Task Par. + Locality



A very similar 'coforall' + 'on' for multilocale parallelism

```
coforall loc in Locales do on loc {
  coforall gpu in here.gpus do on gpu {
    var A: [1..10] int;

    forall elem in A do
      elem += 1;
  }
}
```

Chapel Code: Data + Task Par. + Locality + Data Movement



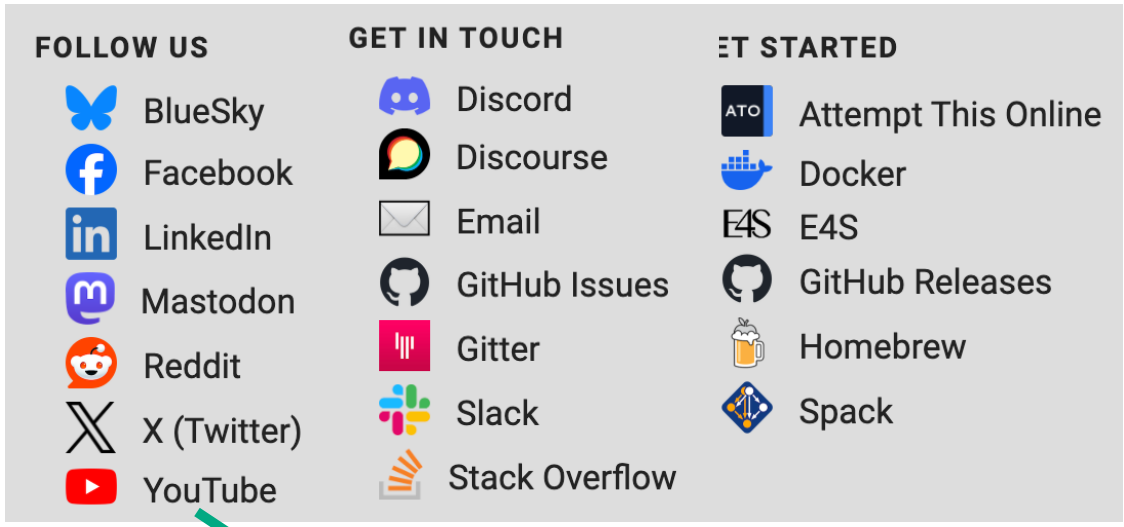

```
var CpuArr: [1..10] int;
coforall loc in Locales do on loc {
  coforall gpu in here.gpus do on gpu {
    const myChunk = start..end; // math omitted
    var A = CpuArr[myChunk];


    forall elem in A do
      elem += 1;
  }
}
```

Arrays or slices can be copied across any locales, including a GPU sublocale

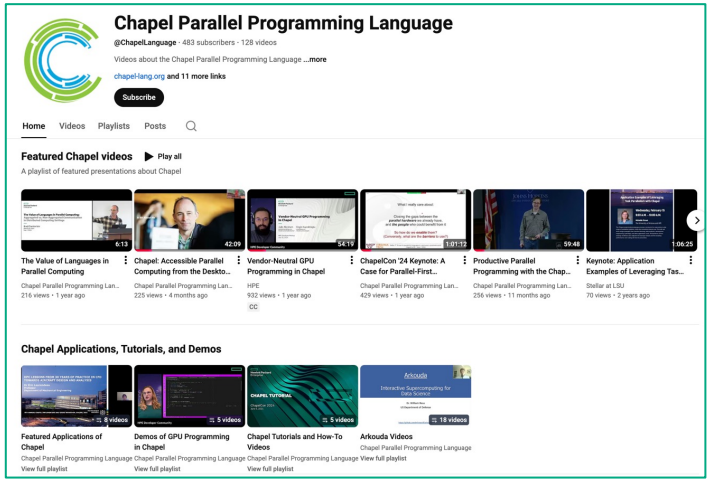
In Closing...

Ways to Engage with the Chapel Community

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



Many learning resources are on YouTube



Ideas/Wishes for Chapel Applications or Libraries?

We are collecting **ideas** to develop more **libraries** & **applications** in Chapel

Ideas for enriching the Chapel ecosystem #28389

e-kayrakli started this conversation in **Ideas**

e-kayrakli on Feb 10 Maintainer edited

In the upcoming months, we (the core development team at HPE) are planning to write more Chapel code ourselves. The goals for this practice is to:

- Create a set of modules that are reusable by others
- Write sample mini-applications that may be specific to a field to showcase Chapel's capabilities
- Discover any potential limitations or issues to help improve the language and its implementation along the way

To that end, in this discussion, I want to start collecting some ideas. Where ideally an idea can come to fruition within order of weeks, including any potential fixes or improvements to Chapel itself. They can be worked on individually or in small groups. The end result could be a first version of a library, or a piece of application that can be open-sourced and advertised/included in Chapel examples.

If you have any ideas from your area of expertise or interest, please drop a comment here. Bonus, if you can point to examples implemented in other languages to provide a source of inspiration.

If you want to start working on any of the ideas, please drop a comment under it to foster collaboration and avoid redundant effort as much as we can.

↑ 1

https://github.com/chapel-lang/chapel/discussions/28389

Closing Remarks

Chapel is a **parallel, scalable, portable, and open-source** programming language

Its users:

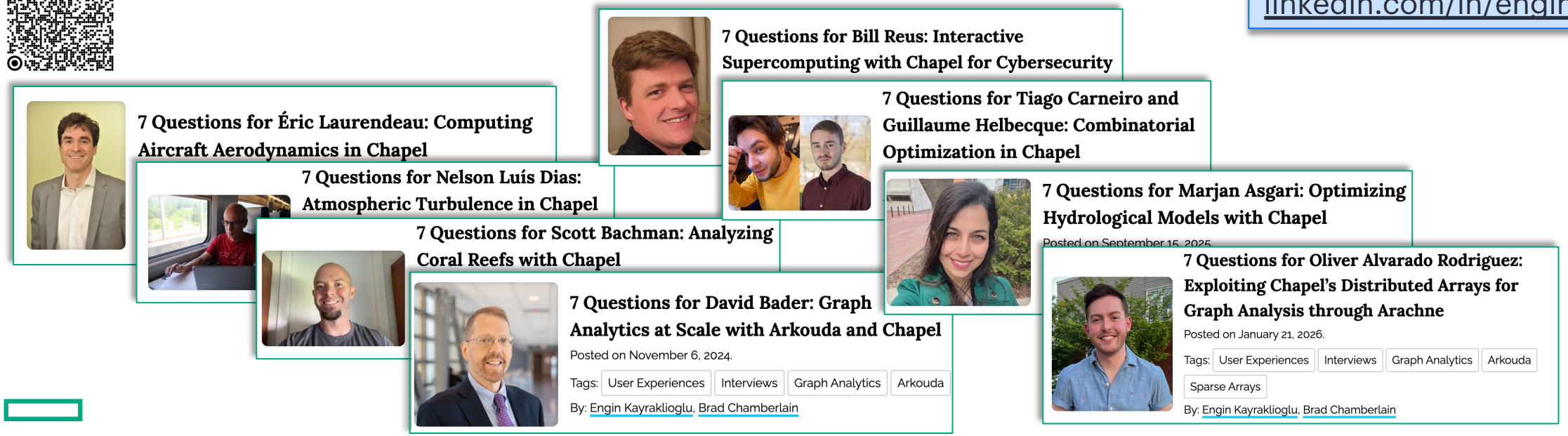
- **easily prototype** their applications on their laptop and **run it on a supercomputer**
- **onboard new developers** with different backgrounds to their teams easily
- use the **same code base on different architectures**

THE LINUX FOUNDATION

HPSF
HIGH PERFORMANCE SOFTWARE FOUNDATION

Read Chapel users' stories on Chapel Blog:

chapel-lang.org/blog/series/7-questions-for-chapel-users/



Contact Info

linkedin.com/in/engink

Thank You