

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...



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

Coral Reef Biodiversity Detection w/ Spectral Image Analysis

1. Read in a (M x N) raster image of habitat data
2. Create a (P x P) mask to find all points within a given radius.
3. Convolve this mask over the entire domain and perform a weighted reduce at each location.

Runs on Frontier!


- 5x improvement going from 2 to 64 nodes (from 16 to 512 GPUs)
- Straightforward code changes:
 - from sequential Chapel code
 - to GPU-enabled one
 - to multi-node, multi-GPU, multi-thread

Read Scott Bachman's interview

7 Questions for Coral Reefs

Posted on October 12, 2021

Tags: Earth Science, User Experiences



Monte Carlo Simulation for Light Transport in Tissue

Problem

- Simulate many many photons going through tissue

Background

- This is a relatively straightforward Monte Carlo simulation
- There's a sequential implementation in pure C

What did we do?

- Transliterate the code from C to Chapel
- Add GPU capabilities

How does it perform?

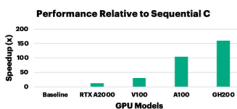
- Up to 160x improvement

Why does it matter?

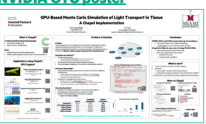
- More readable than the sequential C implementation
- 300 lines of heavily commented Chapel code:

<https://github.com/e-kavrakil/mc321.chpl>

Performance Relative to Sequential C



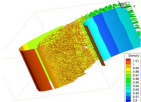
NVIDIA GTC poster



CHAMPS

- **CHApel MultiPhysics Simulation**
- Designed for aircraft aerodynamics
- Can simulate multiple physical phenomena
 - Flow, droplets, icing, turbulence...
- **> 150,000 lines** of Chapel code
- Developed and maintained by mechanical engineering grad students
- Achieves similar fidelity to commercially developed software

High-fidelity Multiphysics Simulation




Prof. Eric Laurendeau's CHIUW 2021 Keynote

CHIUW 2021 Keynote—HPC Lesson Learned: Aerospace Design and Simulation

7 Questions for Aircraft Aerodynamics

Posted on September 12, 2021

Tags: User Experiences



Arkouda


A numpy/pandas inspired Framework

- Primary use case is Exploratory Data Analysis
- Key goal is to be able to interactively explore large datasets

Arkouda is implemented with Chapel

- The user interacts with the framework through a Python interface
- The server is implemented in Chapel

Website: <https://arkouda.readthedocs.io>



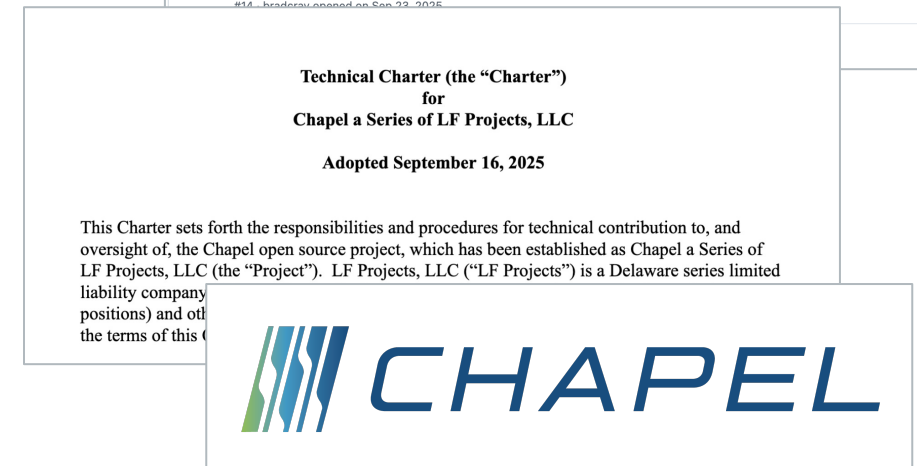
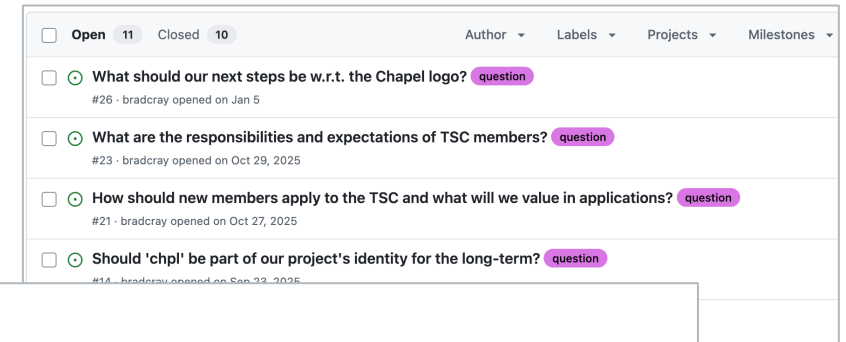
User Experiences Leveraging Chapel's Features for Productive Parallel Programming

Wednesday, 11:15-11:40
Ballroom A-D




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



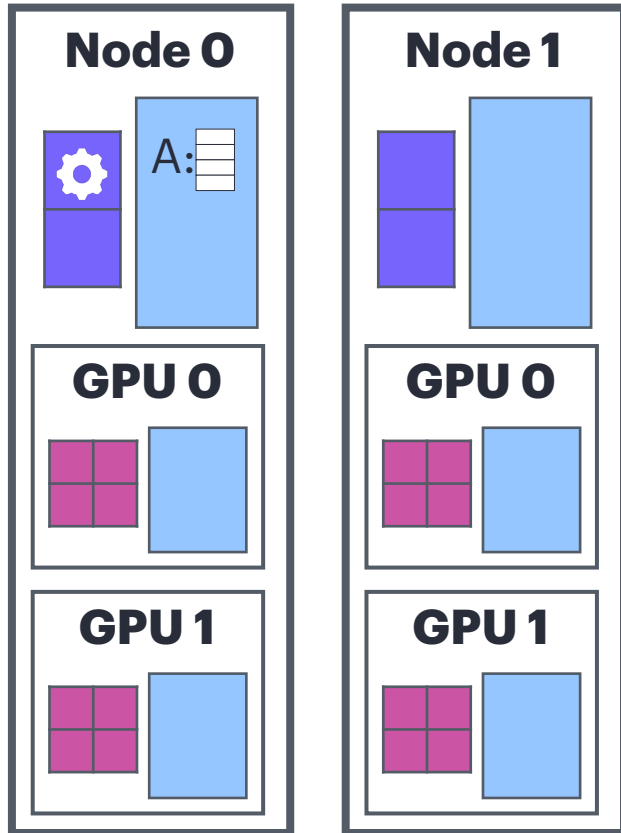
What does Chapel code look like?



Chapel Code: Fundamentals

■ CPU Core ■ GPU Core

■ Memory



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

Local, non-distributed array allocation

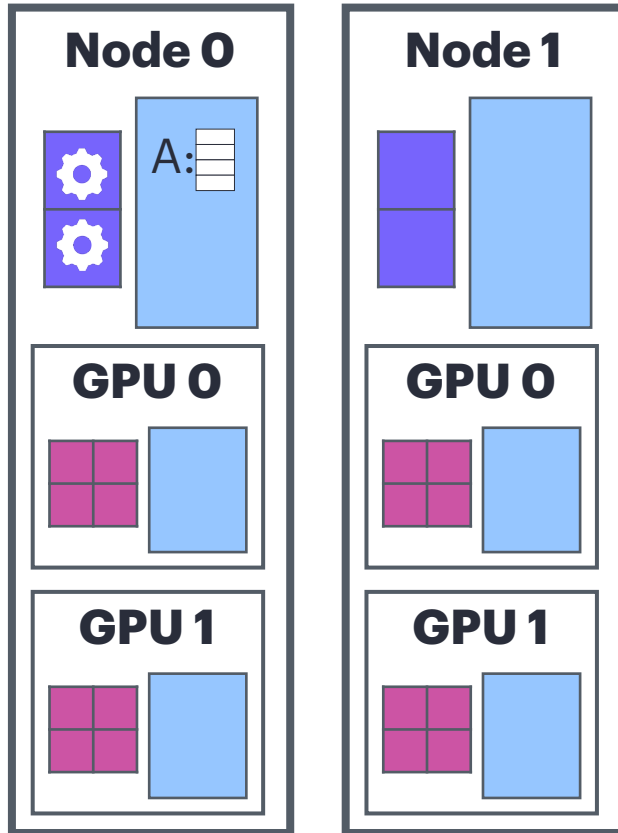
```
for elem in A do  
  elem += 1;
```

Sequential iteration over the array

Chapel Code: Basic Data Parallelism

■ CPU Core ■ GPU Core

■ Memory



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

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

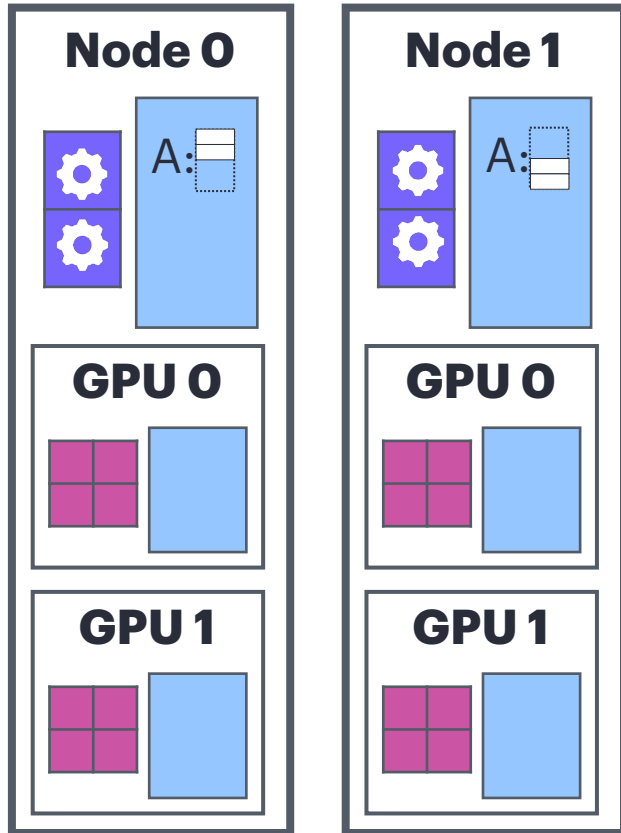
Parallel iteration over the array



Chapel Code: Basic Data Parallelism

■ CPU Core ■ GPU Core

■ Memory



```
use BlockDist;
```

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

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

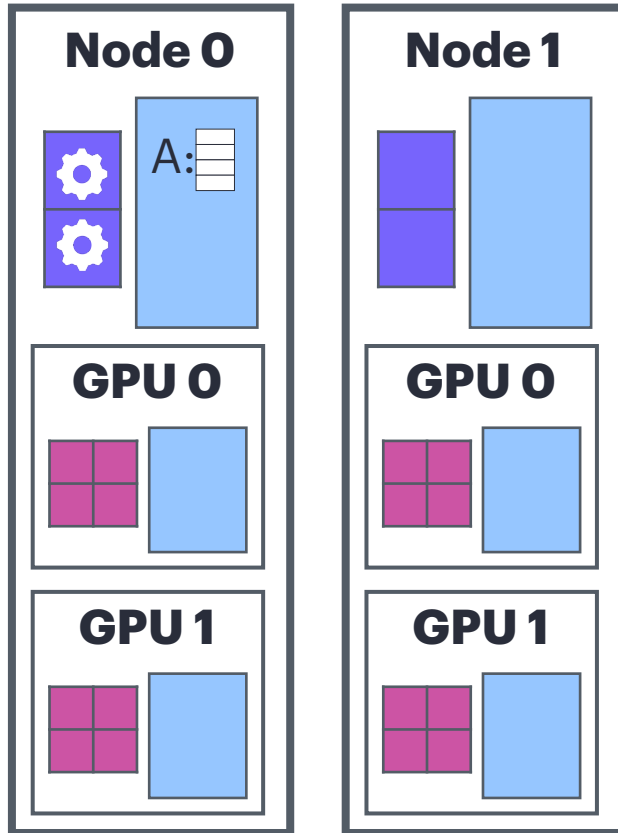
Block-distributed array allocation

Distributed, parallel iteration over the array

Chapel Code: Basic Data Parallelism

■ CPU Core ■ GPU Core

■ Memory



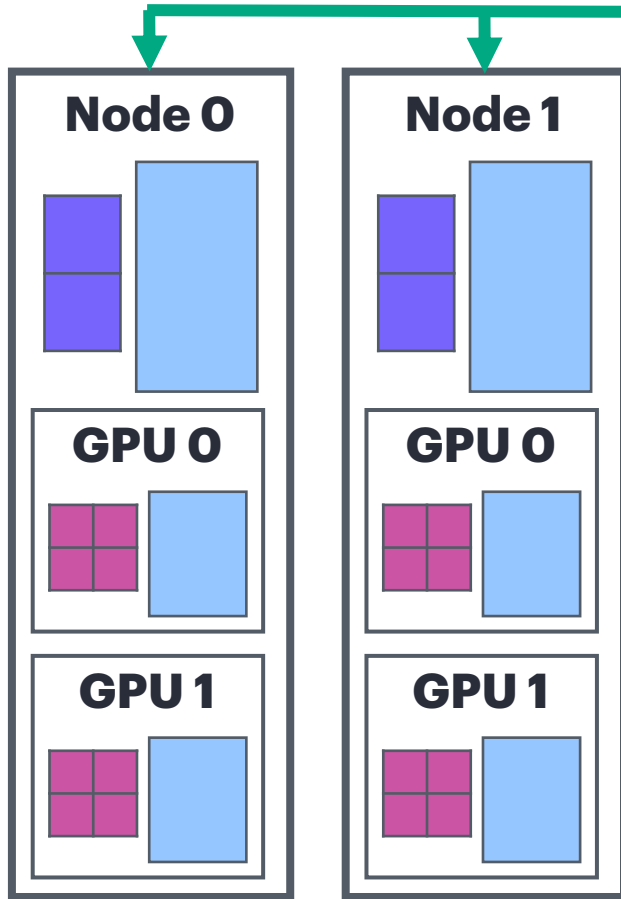
```
var A: [1..10] int;
```

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

Chapel Code: Locality as a First-Class Citizen

■ CPU Core ■ GPU Core

■ Memory



- "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"

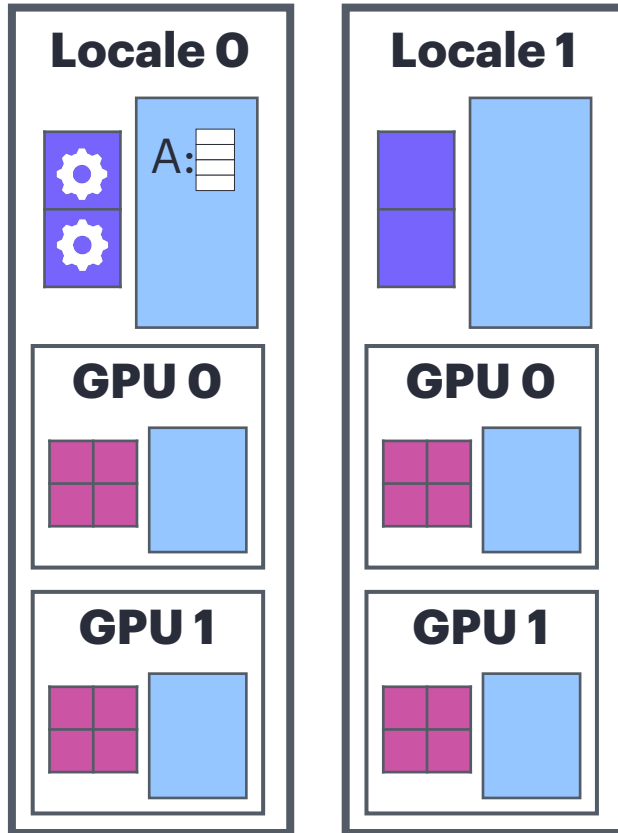
Predefined Variables for Locality

`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

■ CPU Core ■ GPU Core

■ Memory



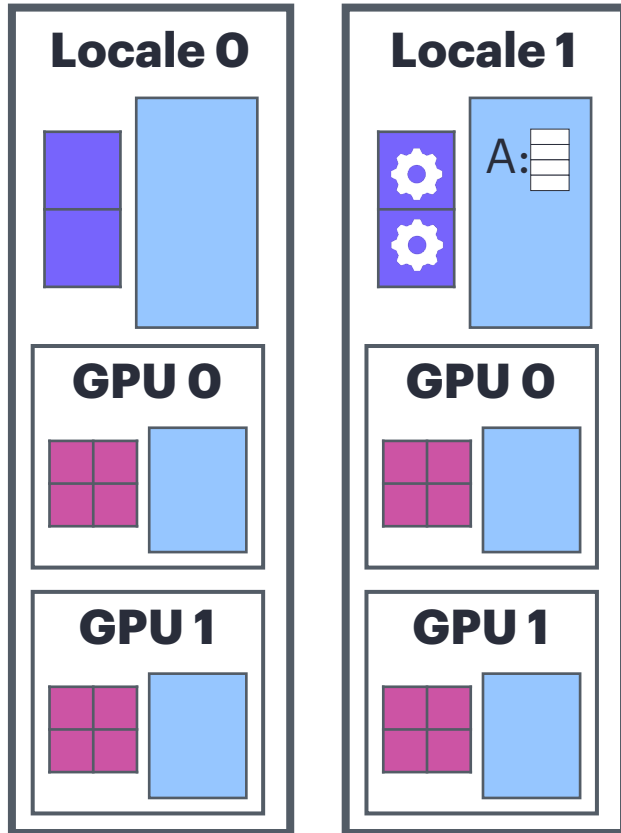
```
var A: [1..10] int;
```

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

Chapel Code: Basic Data Parallelism + Locality

■ CPU Core ■ GPU Core

■ Memory



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

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

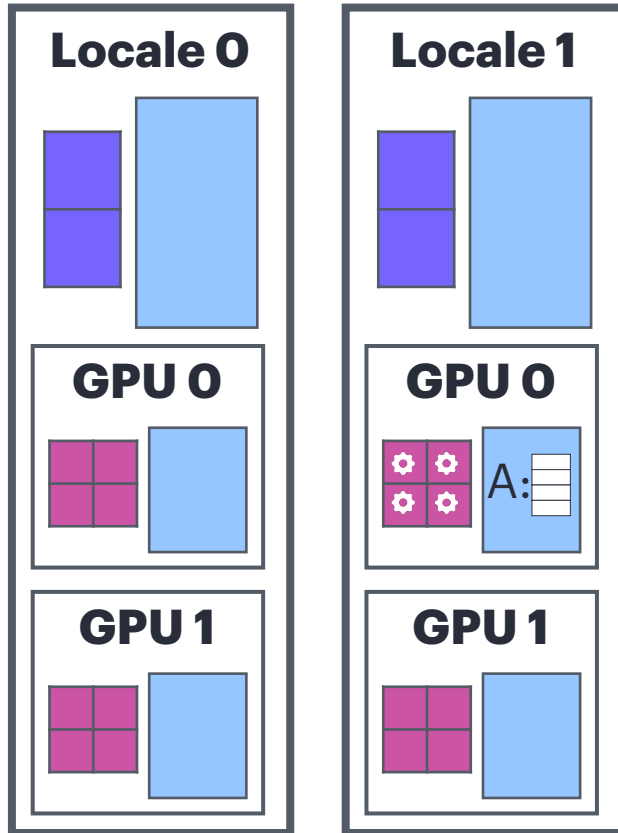
```
}
```

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

Chapel Code: Basic Data Parallelism + Locality

■ CPU Core ■ GPU Core

■ Memory



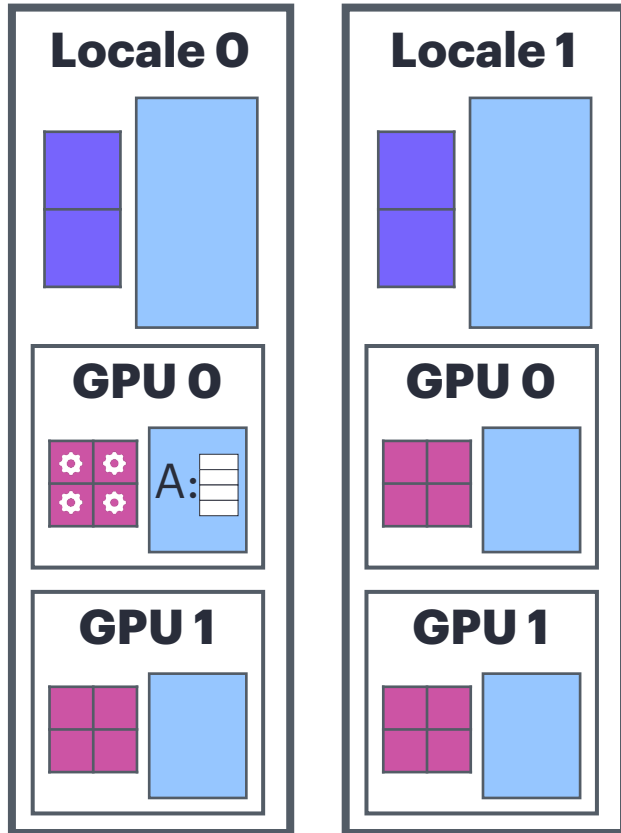
```
on Locales[1].gpus[0] {  
  var A: [1..10] int;  
  
  forall elem in A do  
    elem += 1;  
}
```

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

Chapel Code: Basic Data Parallelism + Locality

■ CPU Core ■ GPU Core

■ Memory



```
on here.gpus[0] {  
  var A: [1..10] int;  
  
  forall elem in A do  
    elem += 1;  
}
```

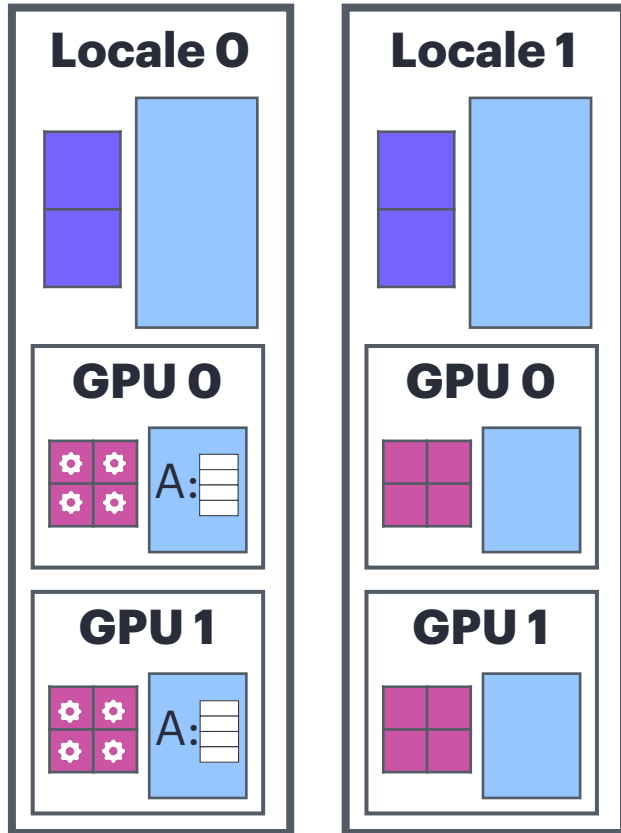
'here' is a built-in representing the current execution locale



Chapel Code: Data + Task Par. + Locality

■ CPU Core ■ GPU Core

■ Memory



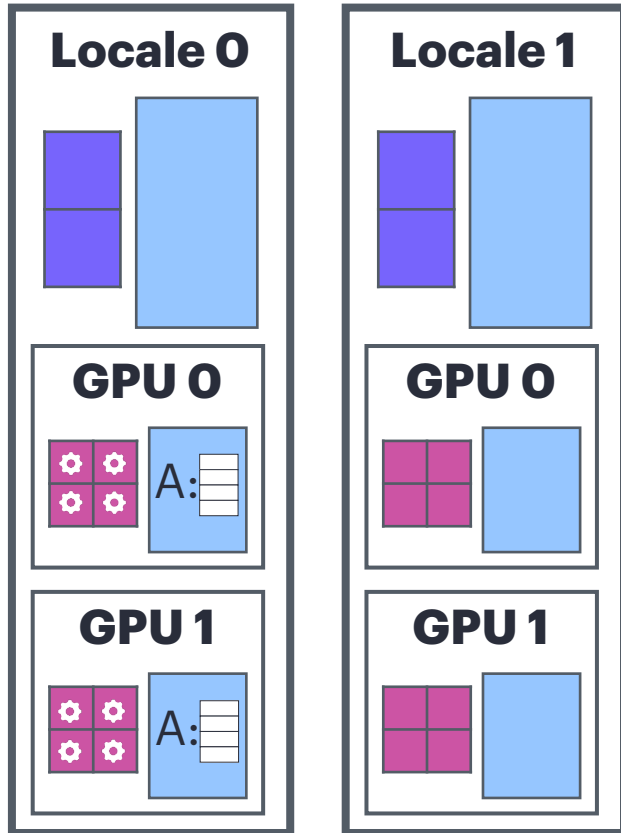
```
coforall gpu in here.gpus {  
  on gpu {  
    var A: [1..10] int;  
  
    forall elem in A do  
      elem += 1;  
  }  
}
```

'coforall' loops run each iteration in a parallel task

Chapel Code: Data + Task Par. + Locality

■ CPU Core ■ GPU Core

■ Memory



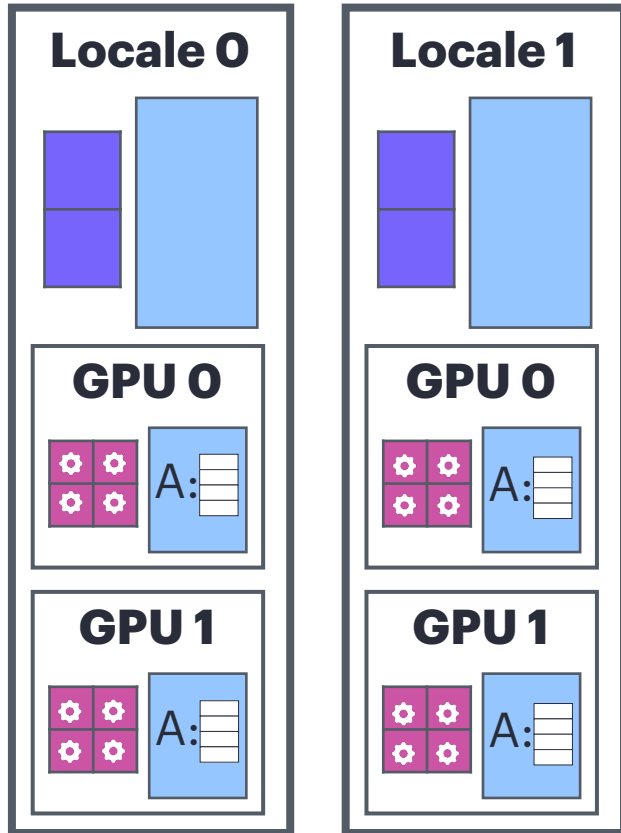
'do' can be used instead of curly braces for single-statement blocks (this is just a matter of style)

```
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

■ CPU Core ■ GPU Core

■ Memory



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;

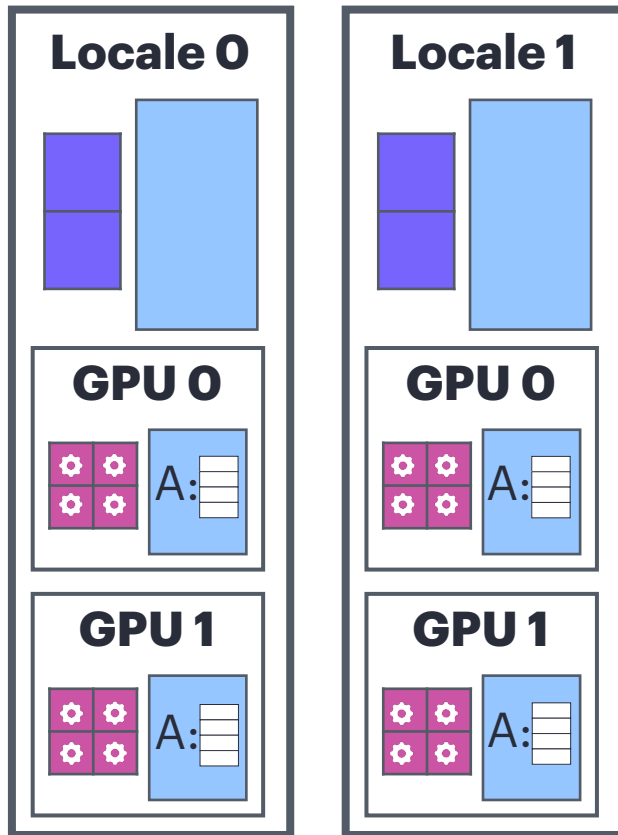
    forall elem in A do
      elem += 1;
  }
}
```



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

■ CPU Core ■ GPU Core

■ Memory



```
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



CHAPEL COMMUNITY

Check out the [Chapel Community Calendar](#) (Downloadable ICS) for a published Outlook calendar of Chapel community events.

Written Q&A / Conversation

Stack Overflow
The ideal place to ask questions that would also benefit others (tagged `chapel`)

Discourse
A place for threaded discussions via email or browser, by category:

- **Announcements:** announcements for the community
- **Newsletters:** quarterly newsletters about Chapel happenings
- **Users:** for questions and discussions from a Chapel user perspective
- **Developers:** for those working on improving Chapel's implementation

Discord
A place for general chat about Chapel

Gitter
Another place for chatting about Chapel

GitHub Issues
The place to file bug reports, feature requests, and other things that deserve action

Live Chapel Events

Chapel project weekly meeting
Our project's main weekly meeting, featuring news, status, demos, discussions, and office hours (Tuesdays @ 10am PT)

Chapel deep-dives and demos
A weekly meeting slot that can be scheduled for an in-depth discussion or demo (Thursdays @ 10am PT)

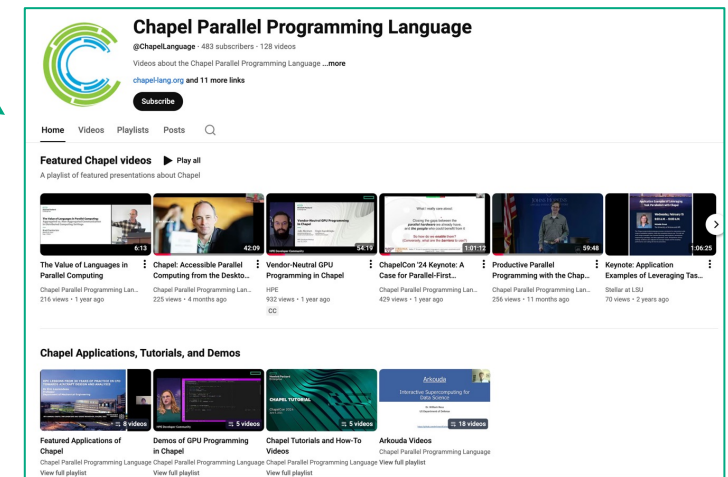
Chapel teaching meet-up
A monthly meeting for those interested in using Chapel in educational settings (2nd Tuesday of each month @ 9am PT)

ChapelCon (formerly CHI UW)
An annual event featuring Chapel-related talks, tutorials, and other sessions

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

FOLLOW US	GET IN TOUCH	GET STARTED
BlueSky	Discord	Attempt This Online
Facebook	Discourse	Docker
LinkedIn	Email	E4S
Mastodon	GitHub Issues	GitHub Releases
Reddit	Gitter	Homebrew
X (Twitter)	Slack	Spack
YouTube	Stack Overflow	

Many learning resources are on YouTube



Chapel Parallel Programming Language
@ChapelLanguage · 483 subscribers · 128 videos
Videos about the Chapel Parallel Programming Language...more
[chapel-lang.org](#) and 11 more links
Subscribe

Home Videos Playlists Posts

Featured Chapel videos ▶ Play all
A playlist of featured presentations about Chapel

The Value of Languages in Parallel Computing · 216 views · 1 year ago

Chapel: Accessible Parallel Computing from the Desktop... · 225 views · 4 months ago

Vendor-Neutral GPU Programming in Chapel · 932 views · 1 year ago

ChapelCon '24 Keynote: A Case for Parallel-First... · 425 views · 1 year ago

Productive Parallel Programming with the Chapel... · 256 views · 11 months ago

Keynote: Application Examples of Leveraging Ter... · 70 views · 2 years ago

Chapel Applications, Tutorials, and Demos

Featured Applications of Chapel · 5 videos

Demos of GPU Programming in Chapel · 2 videos

Chapel Tutorials and How-To Videos · 5 videos

Arkouda Videos · 18 videos

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.



<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**




Read Chapel users' stories on Chapel Blog:



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

Contact Info
linkedin.com/in/engink



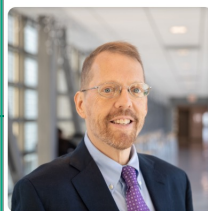
7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel



7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel



7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel



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



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



7 Questions for Marjan Asgari: Optimizing Hydrological Models with Chapel

Posted on September 15, 2025.



7 Questions for Oliver Alvarado Rodriguez: Exploiting Chapel's Distributed Arrays for Graph Analysis through Arachne

Posted on January 21, 2026.

Tags: User Experiences Interviews Graph Analytics Arkouda

Sparse Arrays

By: [Engin Kayraklioglu](#), [Brad Chamberlain](#)



Thank You

