Scalable Parallel Programming with Chapel:
From Multicore CPUs to GPU-Powered Supercomputers

Engin Kayraklioglu, HPE

LUMI User Coffee Breaks

October 15, 2025

HPE

a.k.a.
What Chapel Users Get Done and How

Engin Kayraklioglu, HPE

LUMI User Coffee Breaks

October 15, 2025

What is Chapel?

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...
- **expressive:**
  - object-orientation, memory and type safety, error handling...
- **open-source:**
  - a member of High-Performance Software Foundation & Linux Foundation...

THE LINUX FOUNDATION

HPSF HIGH PERFORMANCE SOFTWARE FOUNDATION

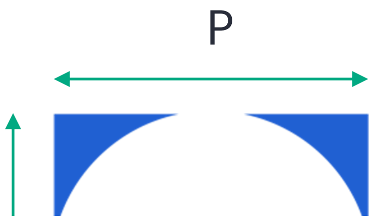
What *do* Chapel users get done?

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.



[Read Scott Bachman's interview](#)

7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel

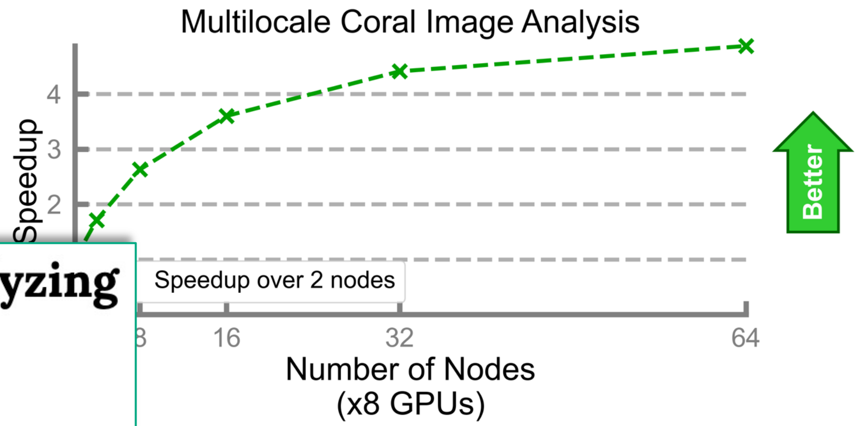
Posted on October 1, 2024.

Tags: Earth Sciences | Image Analysis | GPU Programming | User Experiences | Interviews

- P ~ 400

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

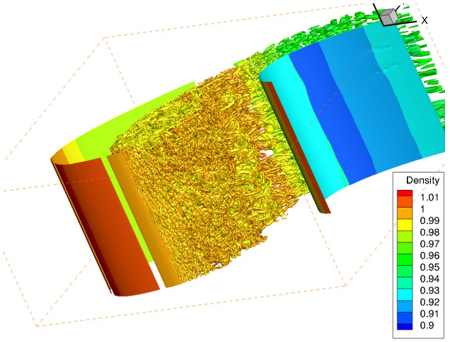


les faster execution with minimal effort

CHAMPS

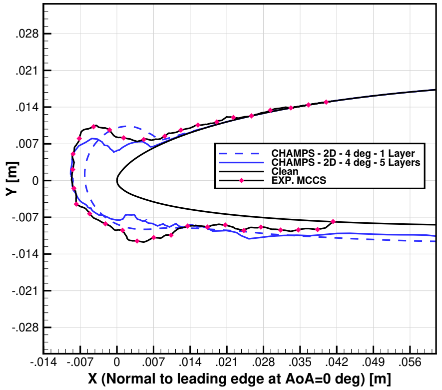
- **CHA**pel **M**ulti**P**hysics **S**imulation
- Designed for aircraft aerodynamics
- Can simulate multiple physical phenomena
  - Flow, droplets, icing, turbulence…

- **> 150,000 lines** of Chapel code
- Developed and maintained by mechanical engineering grad students
- Achieves similar fidelity to commercially developed software

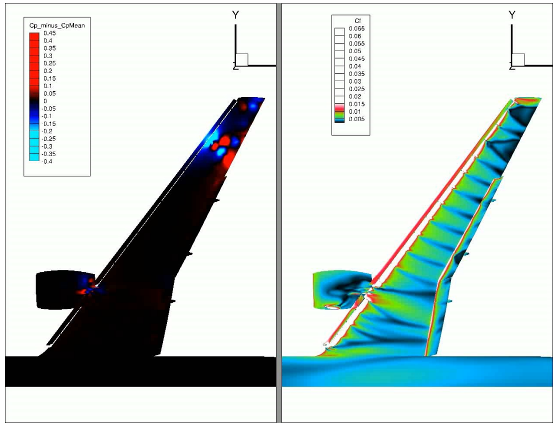


High-fidelity Multiphysics Simulation

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY



Ice Accretion Example



High-lift Simulation

Prof. Eric Laurendeu's [CHIUW 2021 Keynote]

CHIUW 2021 Keynote—HPC Lesso
CFD Towards Aircraft Design and A
Chapel Parallel Programming Language • 452 views
This is the CHIUW 2021 keynote by Éric Laurendeau
Slides available here: https://chapel-lang.org/CHIUW

[His more recent interview]

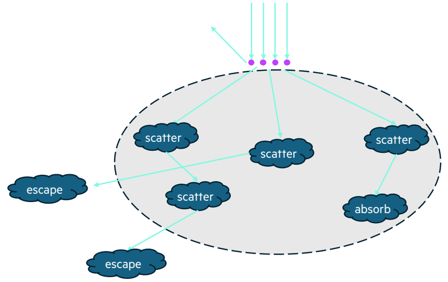
7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel

Posted on September 17, 2024.

Tags: Computational Fluid Dynamics | User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

Monte Carlo Simulation for Light Transport in Tissue



Problem
- Simulate many many photons going through tissue

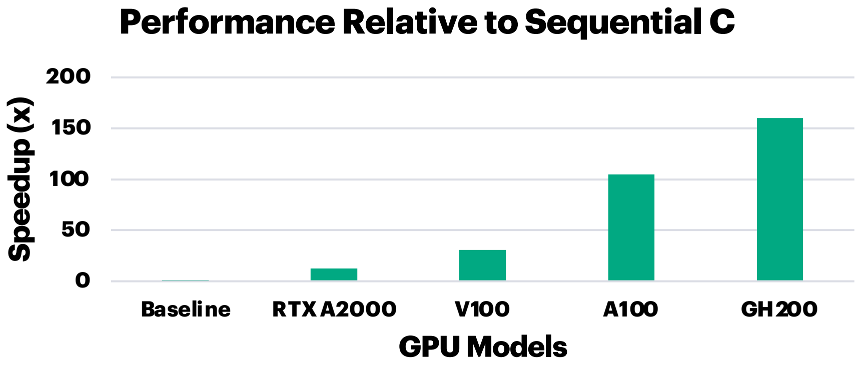
Background
- This is a relatively straightforward Monte Carlo simulation
- There's a sequential implementation in pure C

What did we do?
- Transliterate the code from C to Chapel
- Add GPU capabilities

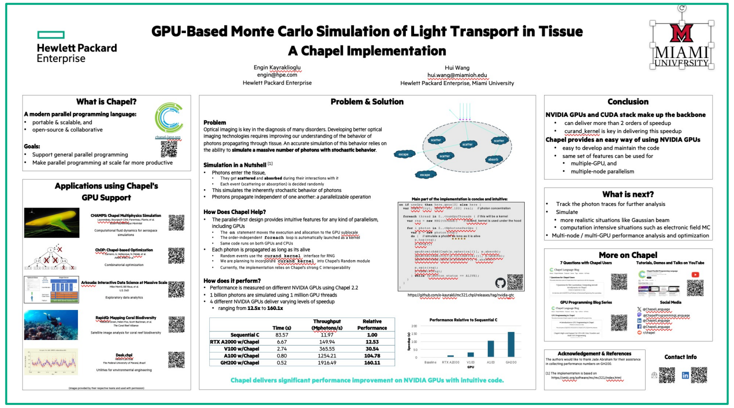
How does it perform?
- Up to 160x improvement

Why does it matter?
- More readable than the sequential C implementation
- 300 lines of heavily commented Chapel code:

  https://github.com/e-kayrakli/mc321.chpl



[NVIDIA GTC poster]



Arkouda

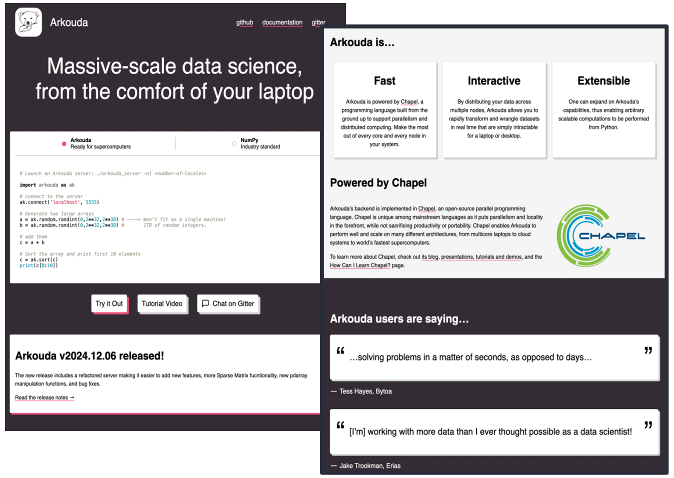
A numpy/pandas inspired Python library

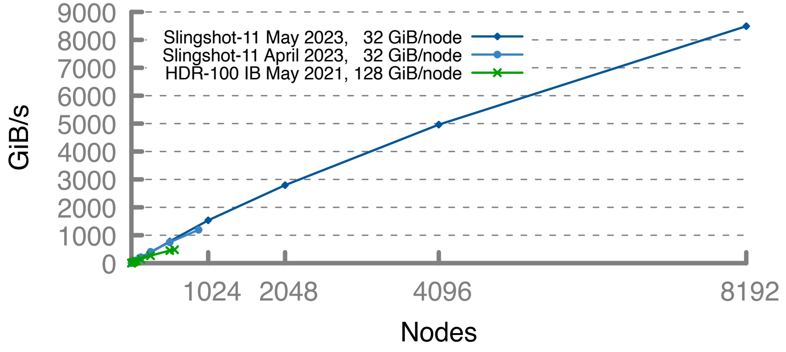
- Primary use case is Exploratory Data Analysis at scale
- Key goal is to be able to interactively analyze terabytes of data

Arkouda is implemented with client/server architecture

- The user interacts with the Python client, likely through Jupyter
- The server is implemented in Chapel, likely running on a supercomputer

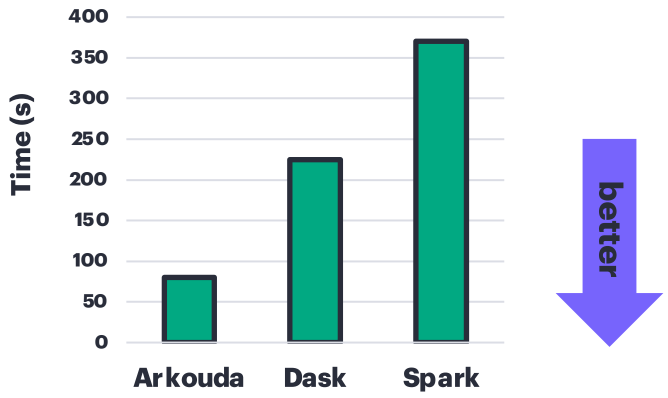
Website: https://arkouda-www.github.io/





Arkouda sorting 256 TiB of data in ~30 seconds

Execution Time
(on 16 HPE Cray EX Nodes)



Arkouda outperforming Dask and Spark in a Server Telemetry Analysis Workflow

Applications Recap

Set of those four Chapel applications:

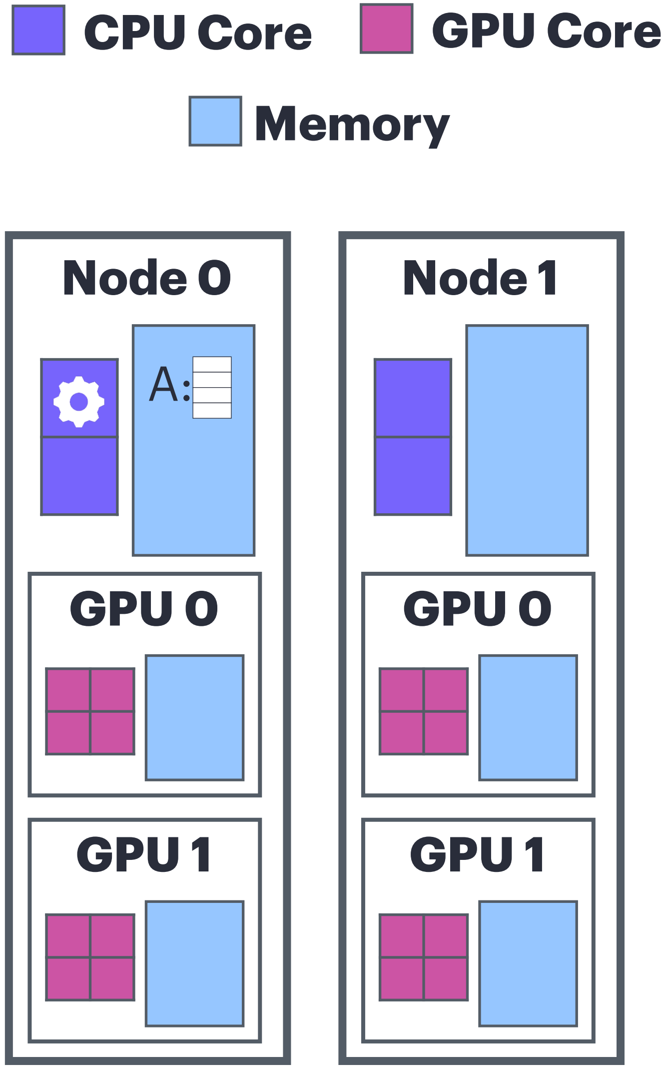
- run on multicore desktops, GPUs, InfiniBand clusters, world's fastest supercomputers
- developed by people from a wide range of backgrounds

All applications use the same language with no extensions etc.

How can Chapel be so versatile?

What does Chapel code look like?

Chapel Code: Fundamentals

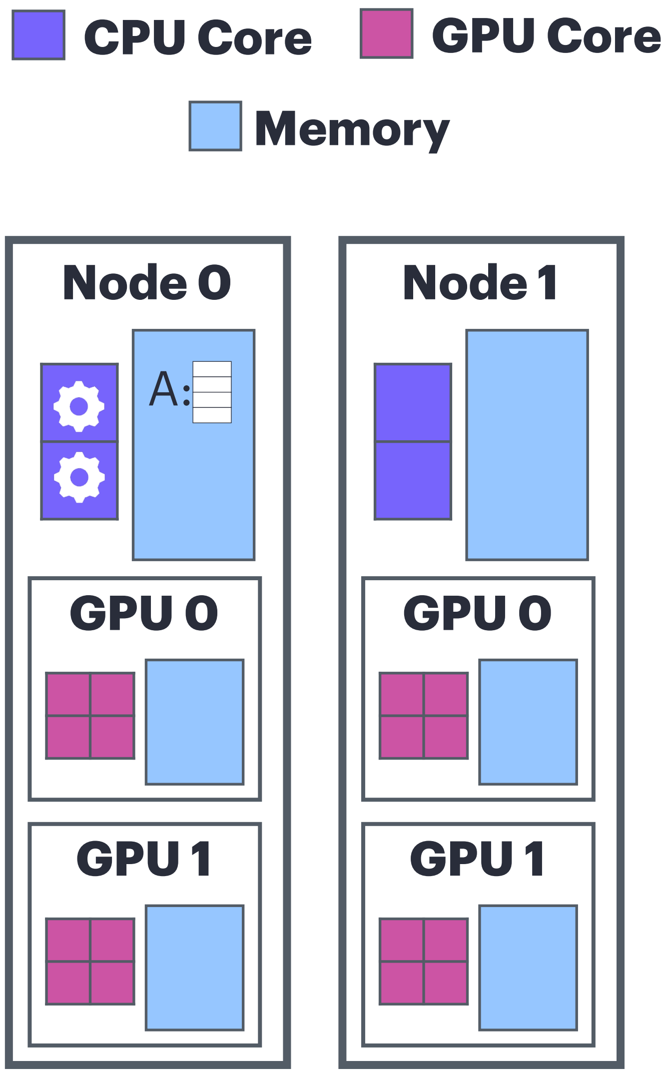



```
var A: [1..10] int;

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

Local, non-distributed array allocation

Sequential iteration over the array

Chapel Code: Basic Data Parallelism



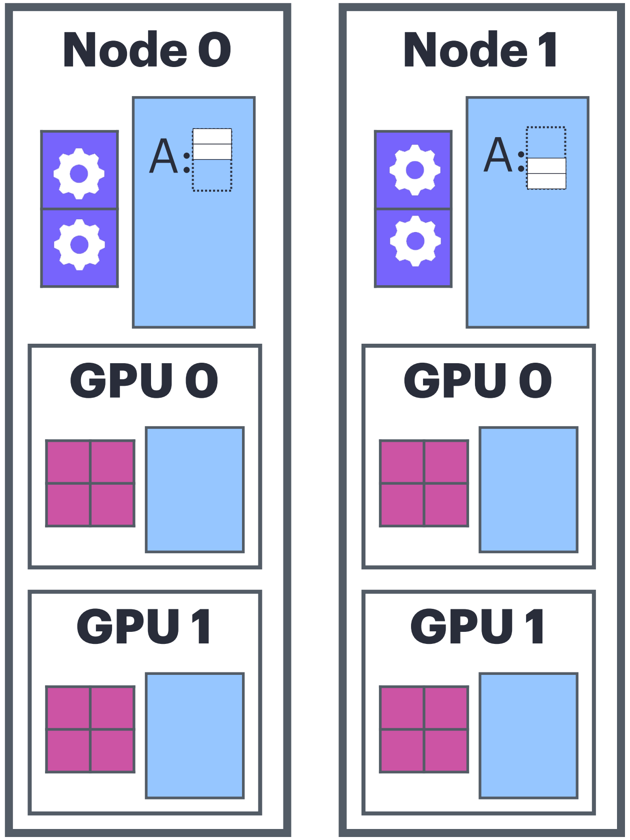

```
var A: [1..10] int;

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

Parallel iteration over the array

Chapel Code: Basic Data Parallelism

CPU Core GPU Core

Memory

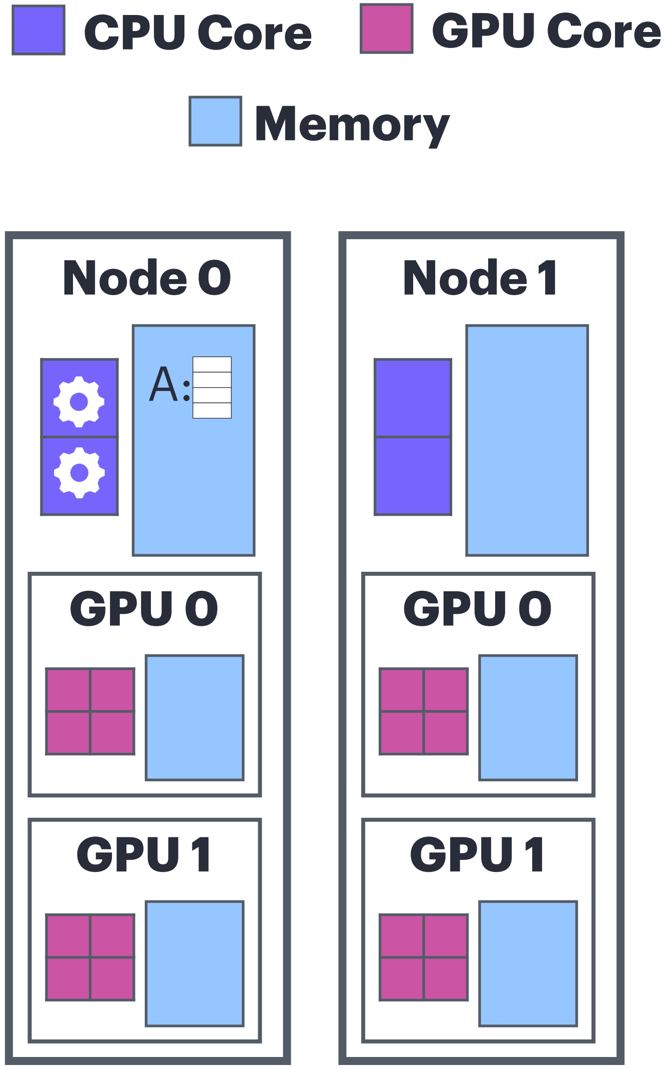


Block-distributed array allocation

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


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

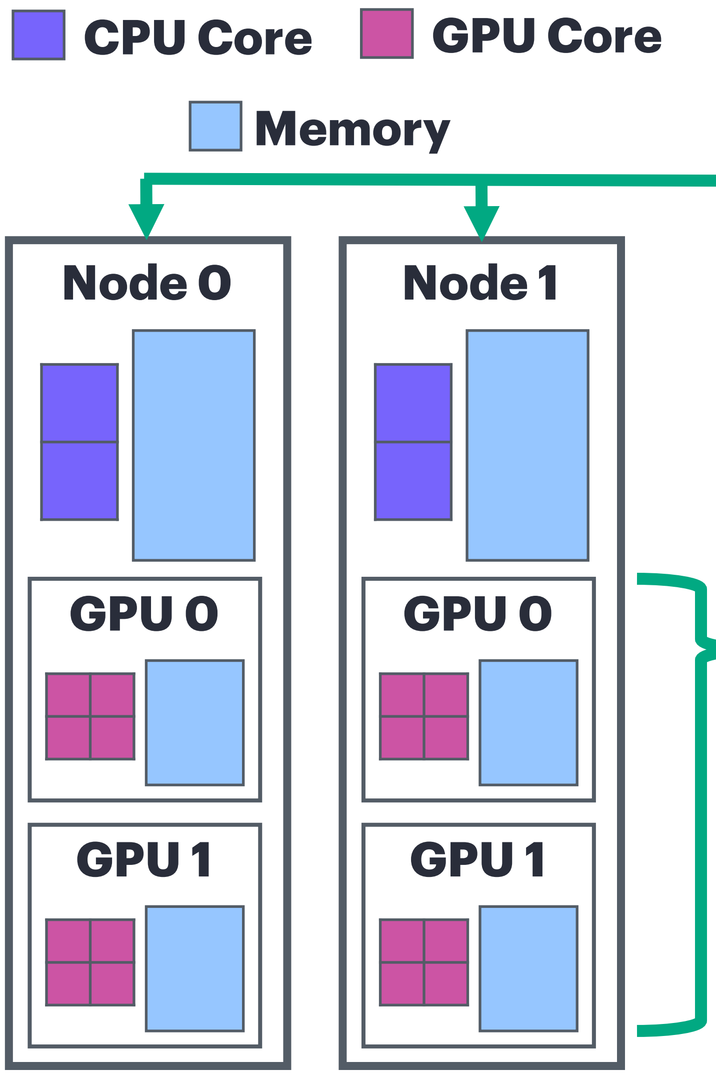
Distributed, parallel iteration over the array

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



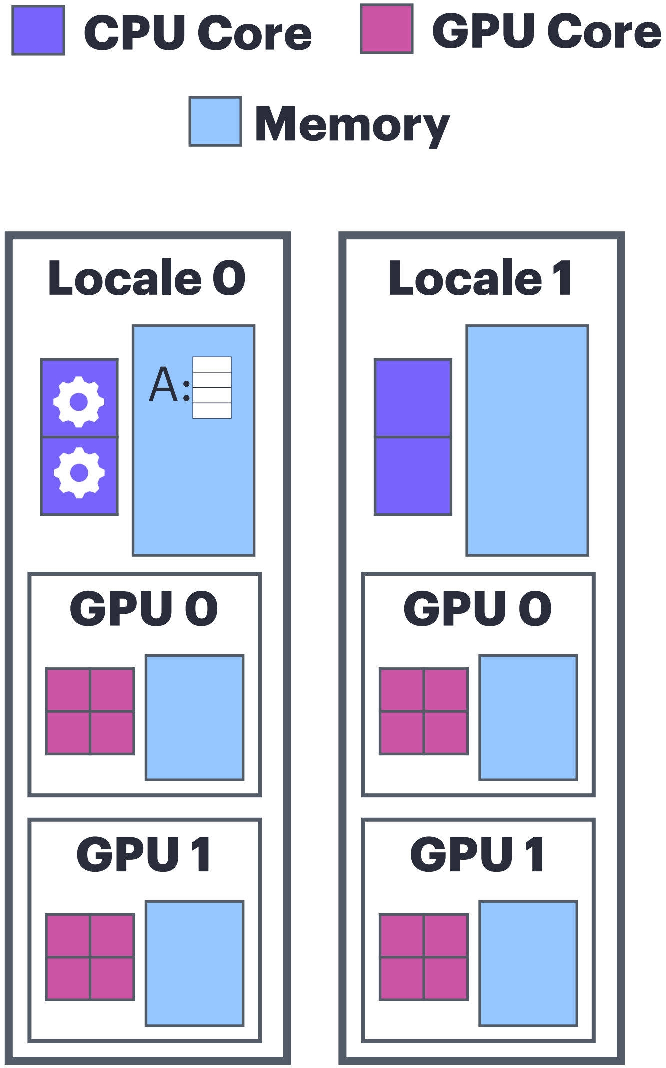
- "locale" is a built-in Chapel type
- In the most common scenario: locale == node
- Under the hood, a locale is always a process
- Users can configure their applications to use:
  - a locale per socket, GPU, NUMA domain...

- GPUs are represented as "sublocales"

Predefined Variables for Locality

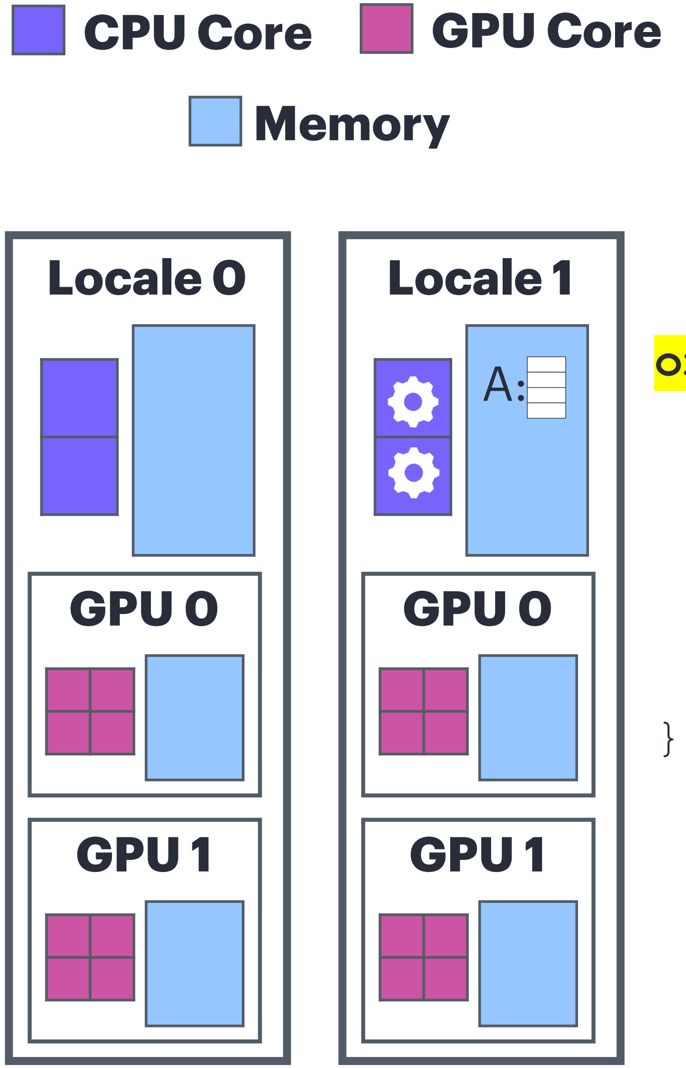
`here`	The current execution locale
`Locales`	An array storing all locales
`locale.gpus`	An array storing all GPU sublocales

Chapel Code: Basic Data Parallelism + Locality




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

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

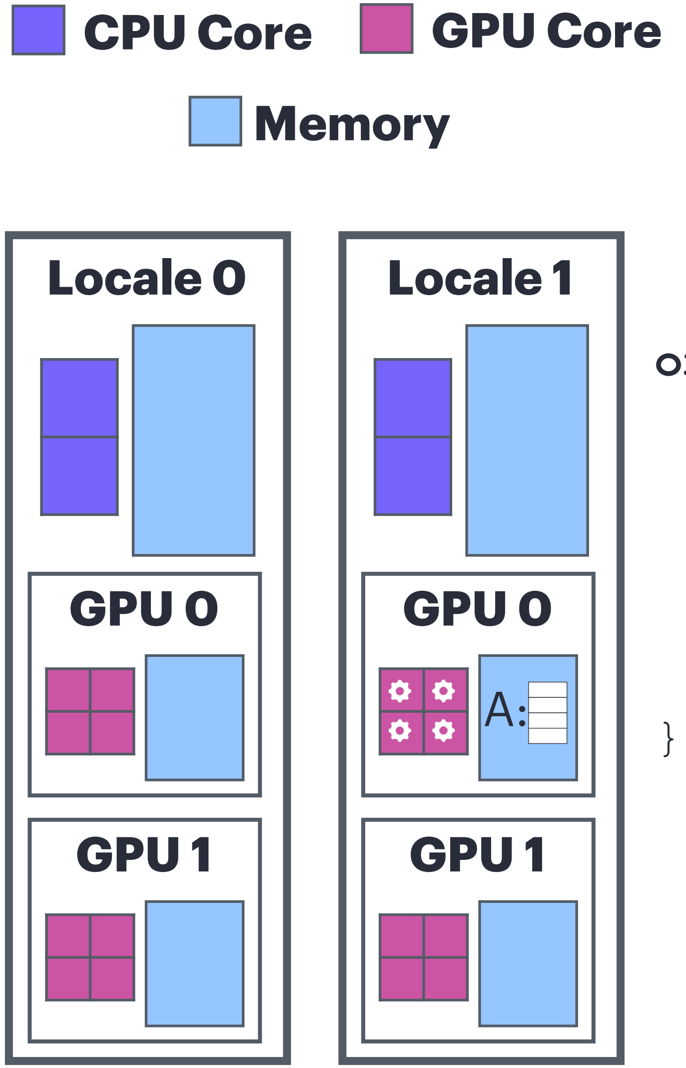
Chapel Code: Basic Data Parallelism + Locality




```
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

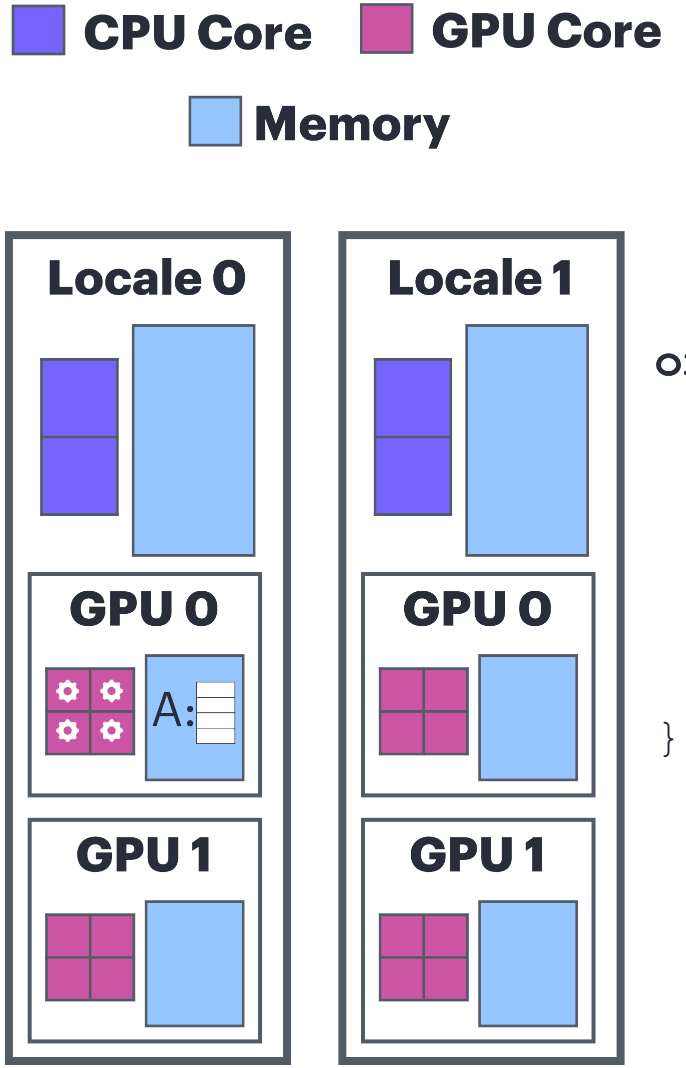
Chapel Code: Basic Data Parallelism + Locality




```
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

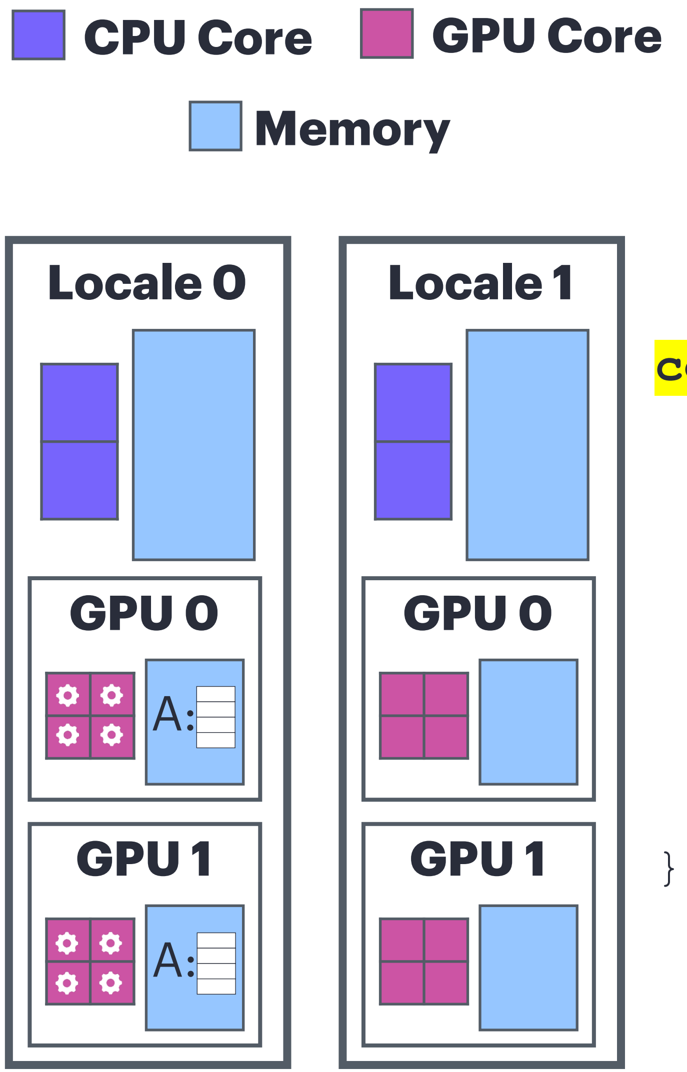
Chapel Code: Basic Data Parallelism + Locality




```
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

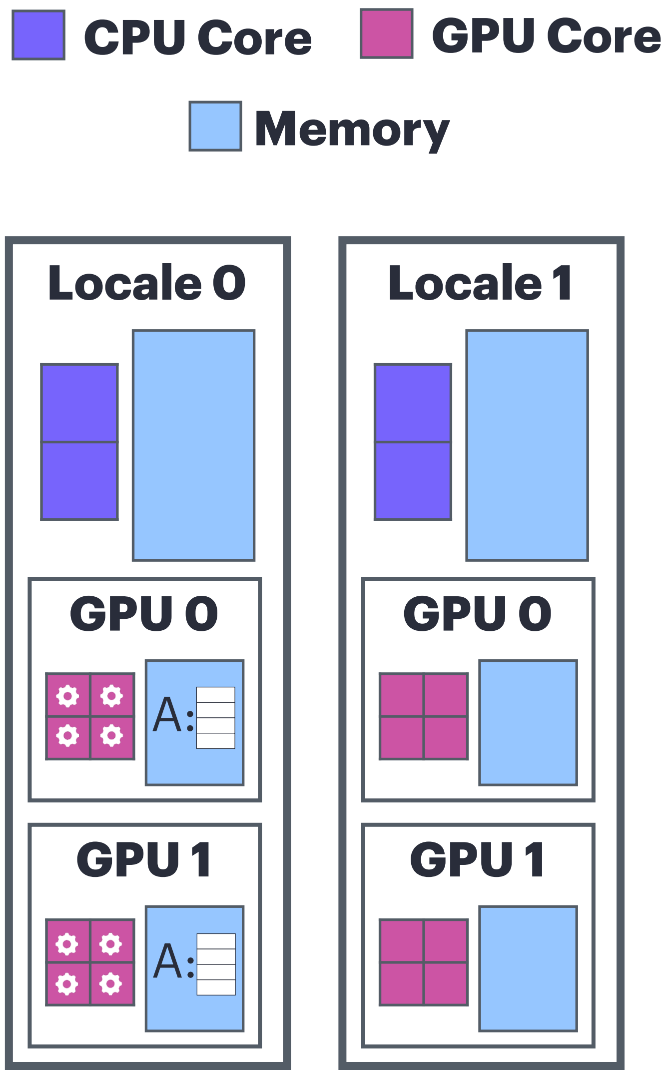
Chapel Code: Data + Task Par. + Locality



'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;
  }
}
```

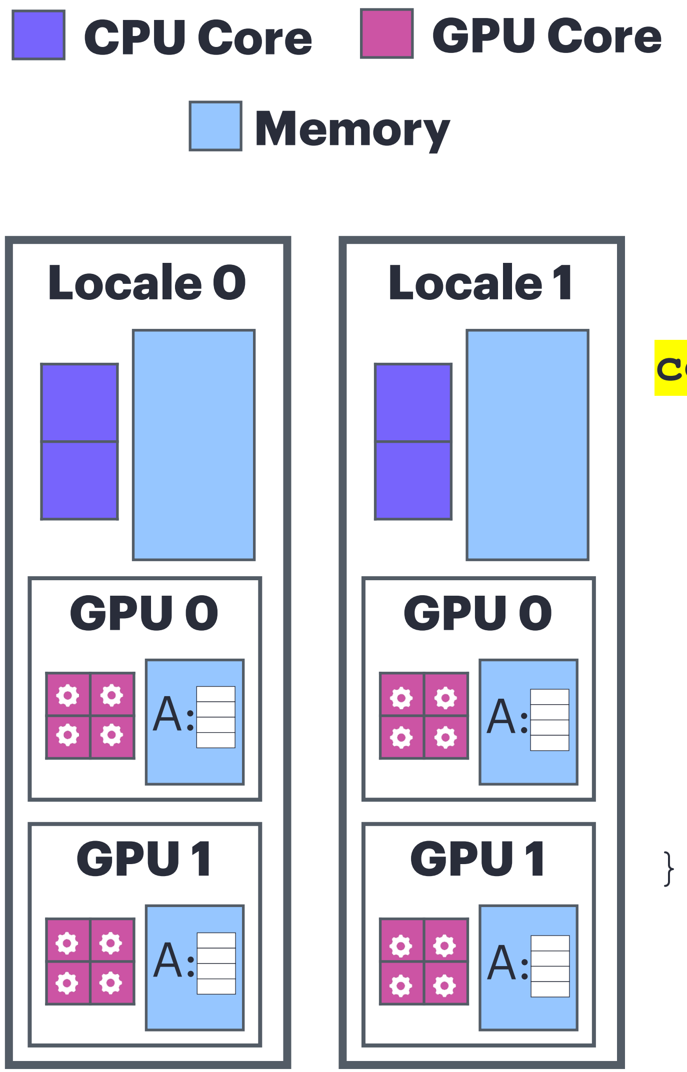
Chapel Code: Data + Task Par. + Locality



'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;
}
```

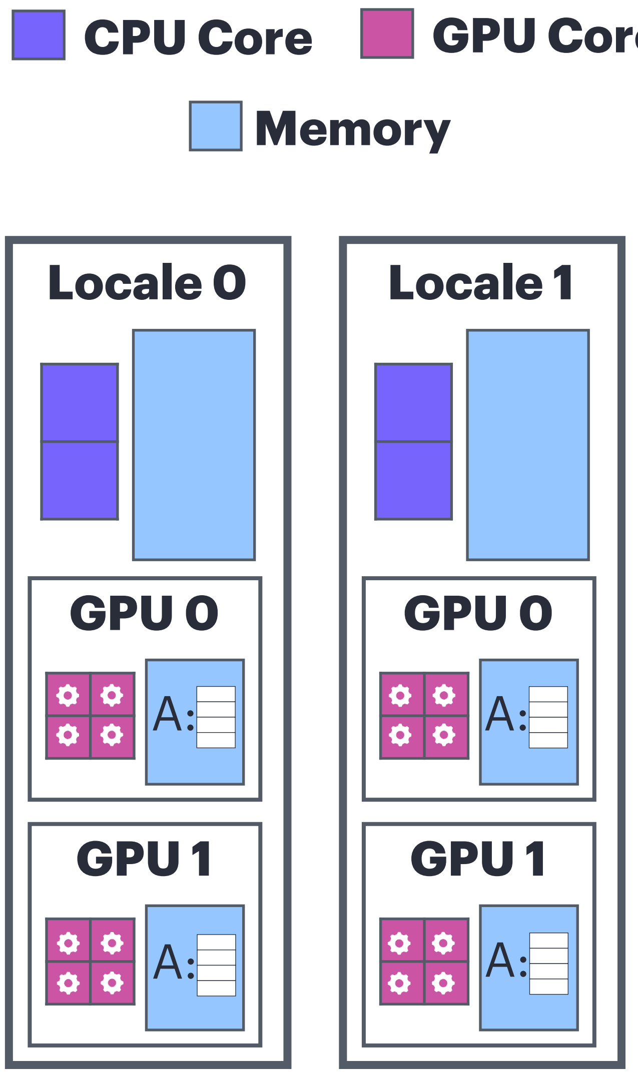
Chapel Code: Data + Task Par. + Locality




```
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;
  }
}
```

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

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

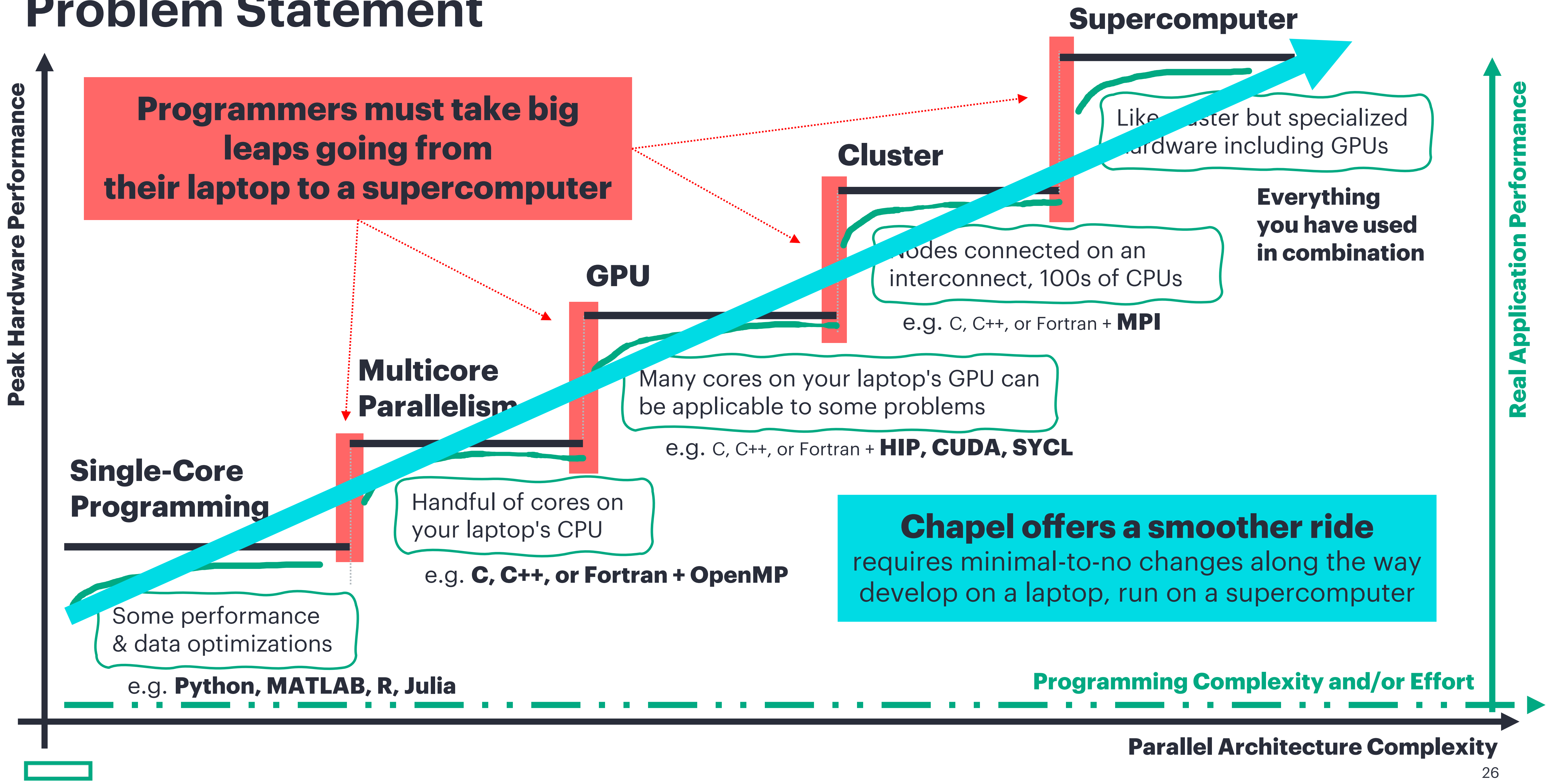



```
var CpuArr: [1..10] int;
coforall loc in Locales do on loc {
  coforall gpu in here.gpus do on gpu {
    const myChunk = start..end; // math omitted
    var A = CpuArr[myChunk];


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

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

The Long Overdue Problem Statement

Problem Statement



Ways to Engage with the Chapel Community


DOWNLOAD DOCS LEARN RESOURCES COMMUNITY BLOG

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**)

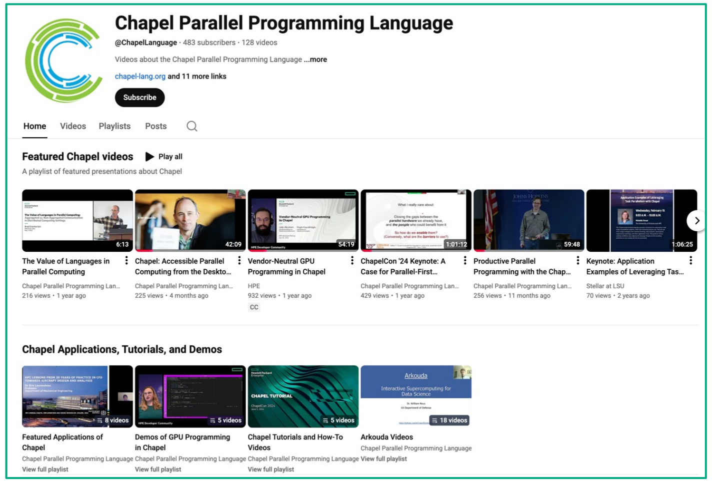
ChapelCon (formerly CHIUW)

An annual event featuring Chapel-related talks, tutorials, and other sesions

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)	Stack Overflow	Spack
YouTube		

Many learning resources are on YouTube



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/


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


7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel


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


7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel


7 Questions for Marjan Asgari: Optimizing Hydrological Models with 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

Contact Info

engin@hpe.com

linkedin.com/in/engink

Next: A success story from LUMI??

Thank You

© 2025 Hewlett Packard Enterprise Development LP