Hewlett Packard
Enterprise

Productive, Vendor-Neutral GPU Programming Using Chapel

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

WACCPD @SC24
November 18, 2024

What is Chapel?

Chapel: A modern parallel programming language

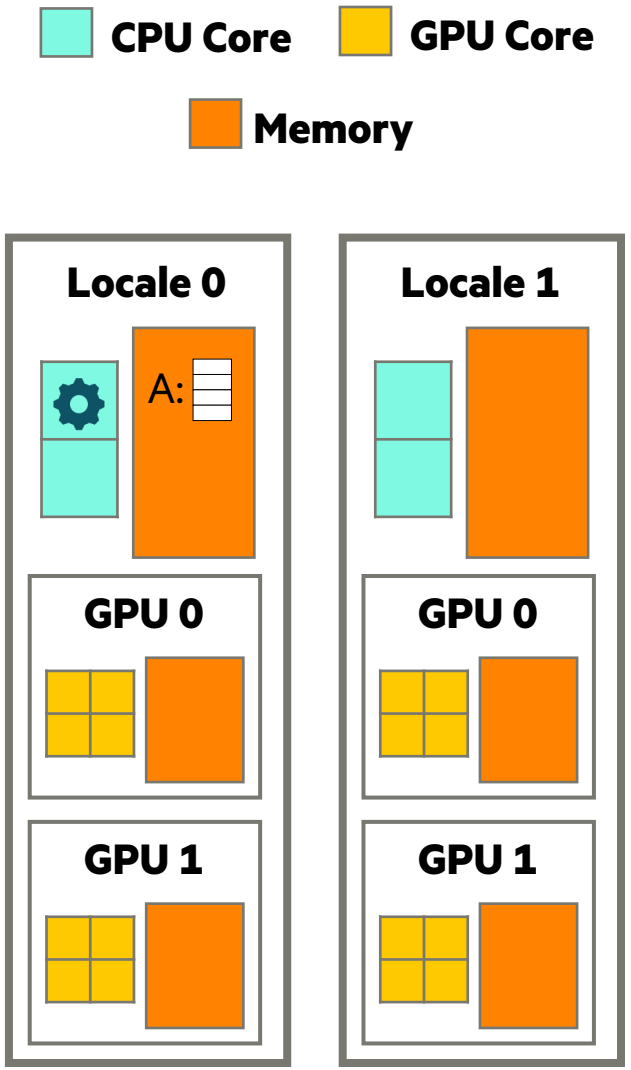
- portable & scalable
- open-source & collaborative

chapel-lang.org

Goals:

- Support general parallel programming
- Make parallel programming at scale far more productive

Programming GPUs with Chapel



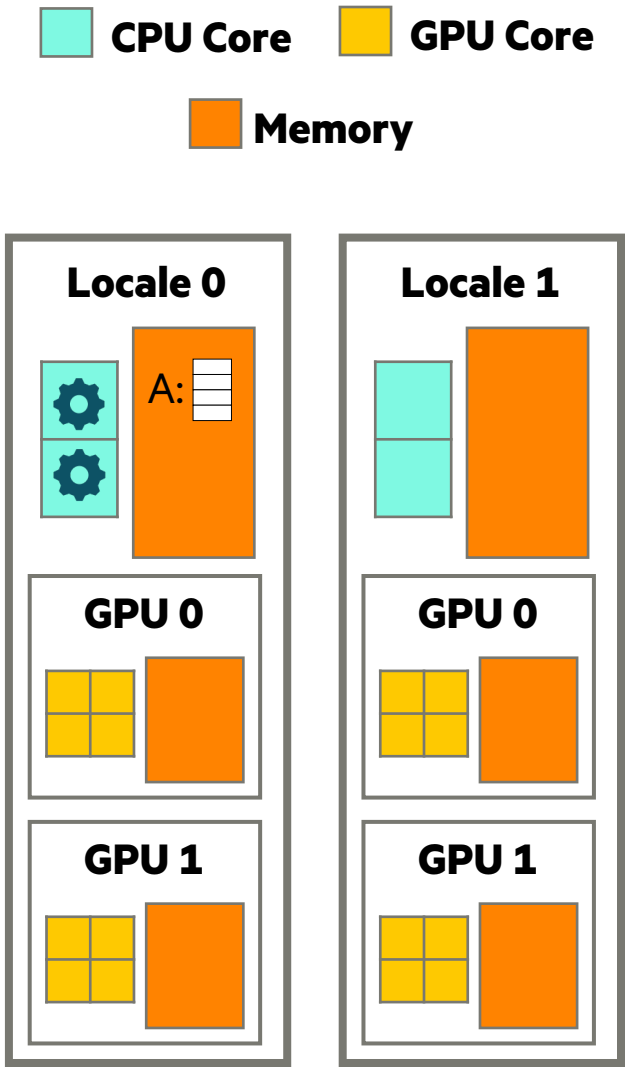

```
var A: [1..10] int;
```

Local, non-distributed array allocation

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

Sequential iteration over the array

Programming GPUs with Chapel

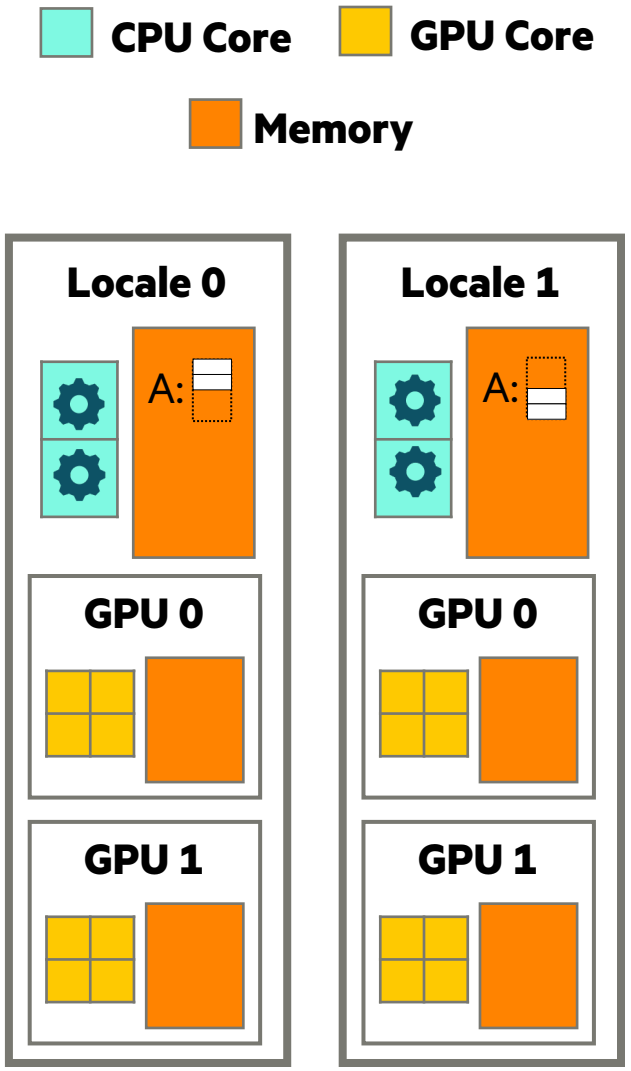



```
var A: [1..10] int;

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

Parallel iteration over the array

Programming GPUs with Chapel



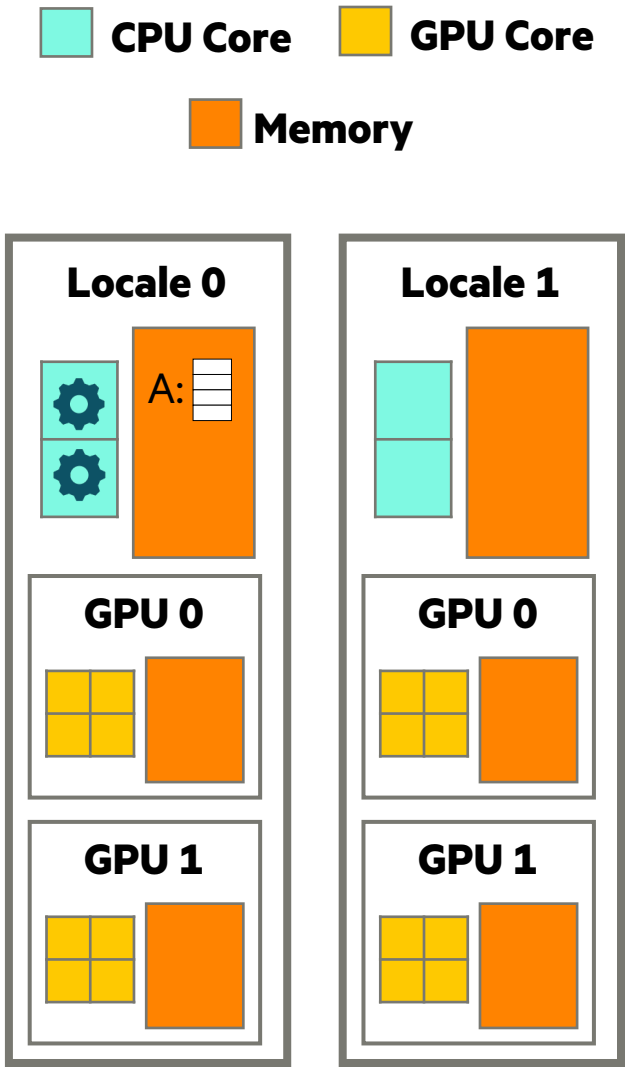

```
use BlockDist;
var Arr = blockDist.createArray(1..10, int);
```

Block-distributed array allocation

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

Distributed, parallel iteration over the array

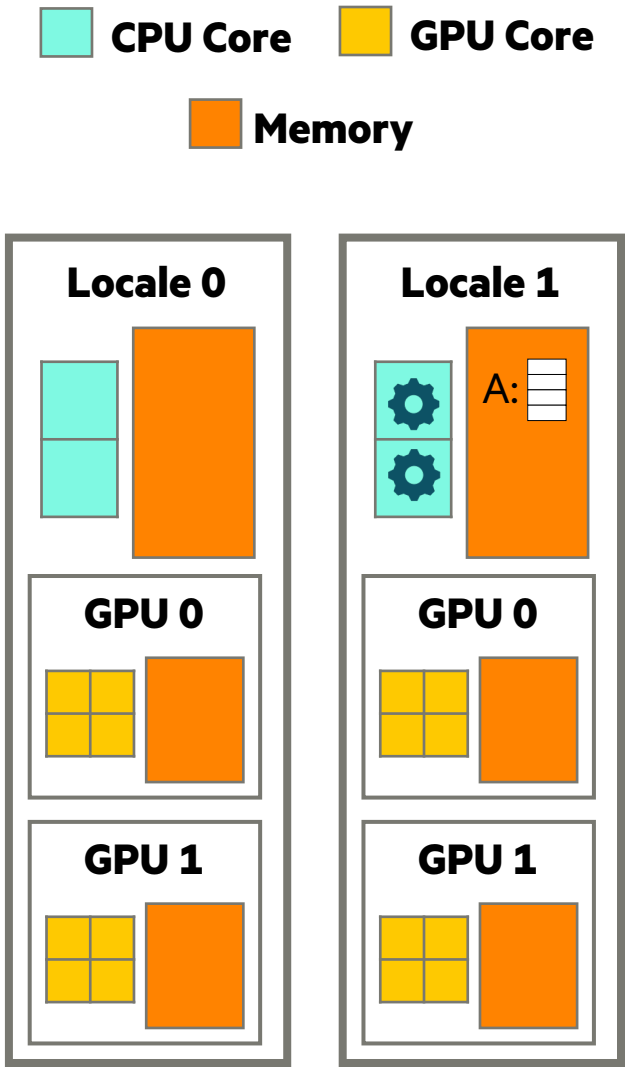
Programming GPUs with Chapel




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

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

Programming GPUs with Chapel

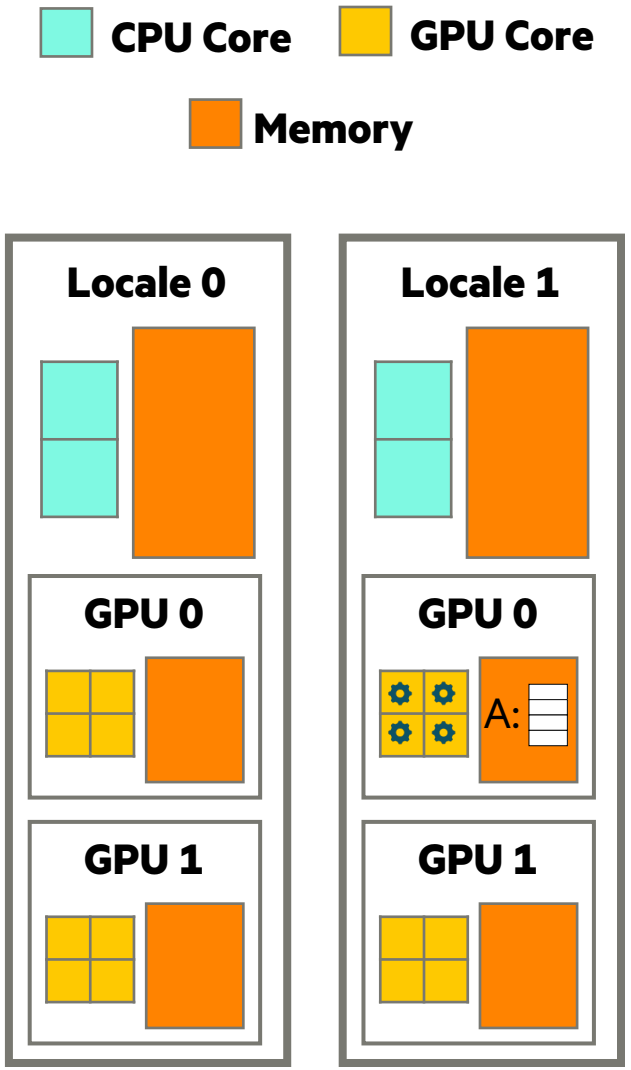



```
on Locales[1] {
  var A: [1..10] int;

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

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

Programming GPUs with Chapel

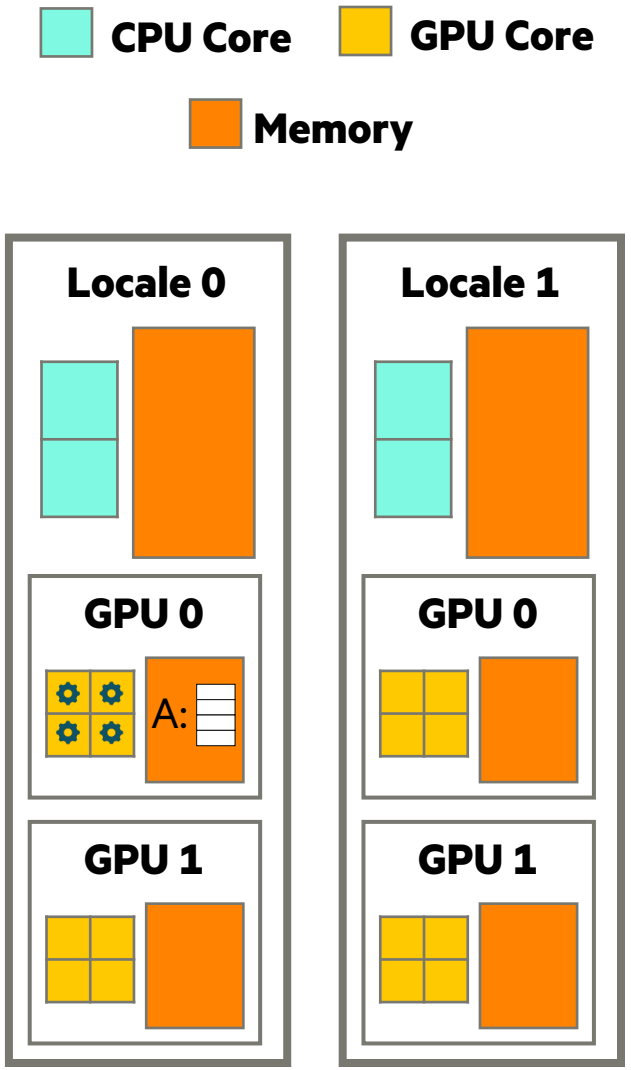



```
on Locales[1].gpus[0] {
  var A: [1..10] int;

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

Each locale object has a 'gpus' array that store GPU sublocales

Programming GPUs with Chapel



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

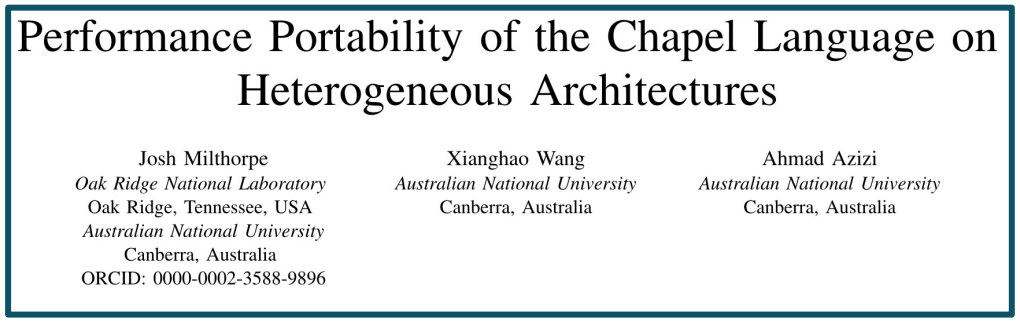
```
on here.gpus[0] {
  var A: [1..10] int;

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

Frequently Asked Questions

- Using distributed arrays to distribute data on multiple GPUs is an active work area
- GPUs are supported only with the LLVM backend, which is the default
  - Chapel can also use C backend
- NVIDIA and AMD GPUs are supported with no special code needed from the user
  - We are on holding pattern to add Intel support
- How does the performance compare?
  - TL;DR Comparable to other technologies, with some exceptions, which we are aware

Milthorpe et al. IPDPSW 2024



Performance Portability of the Chapel Language on Heterogeneous Architectures

Josh Milthorpe
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
Australian National University
Canberra, Australia
ORCID: 0000-0002-3588-9896

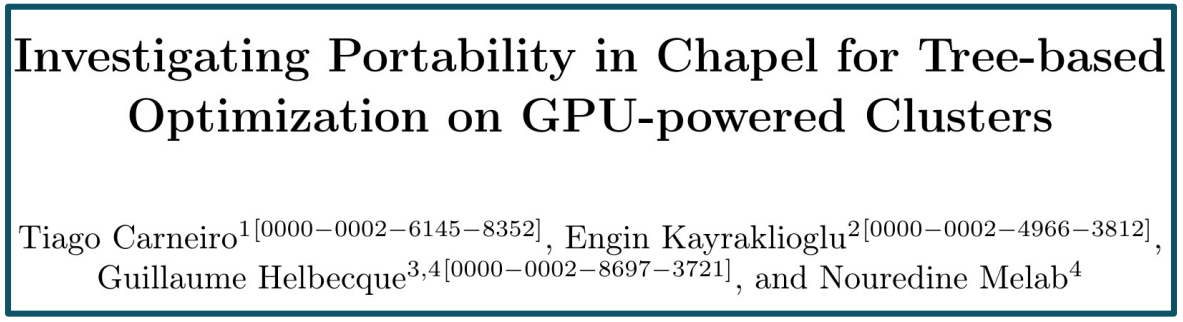
Xianghao Wang
Australian National University
Canberra, Australia

Ahmad Azizi
Australian National University
Canberra, Australia

Using single GPU, compares against

CUDA, HIP, OpenMP, Kokkos

Carneiro et al. Euro-Par 2024



Investigating Portability in Chapel for Tree-based Optimization on GPU-powered Clusters

Tiago Carneiro[1][0000−0002−6145−8352], Engin Kayraklioglu[2][0000−0002−4966−3812], Guillaume Helbecque[3,4][0000−0002−8697−3721], and Nouredine Melab[4]

Using Frontier and Perlmutter, compares against

CUDA, HIP

Learn More

Meet us at the HPE Booth (2219)

Read blog articles

chapel-lang.org/blog/series/gpu-programming-in-chapel/

chapel-lang.org

Watch a Chapel+GPU tutorial

youtube.com/watch?v=1gMFtJN-4_E

Watch a hands-on demo

youtube.com/watch?v=5OqjQhfGKes

Watch a talk+demo

youtube.com/watch?v=nj-WqhGEy24

Hewlett Packard
Enterprise

Thank you!

engin@hpe.com

November 18, 2024