OCT 2023 CHAPEL TUTORIAL
UPDATED FOR USE IN SPRING 2025 CS372 CLASS

Chapel Team, edited by Michelle Strout

April 1, 2025

PLAN

- **Announcements**
  - Final project assignments coming out ASAP
  - SA7 will be posted Wednesday April 2nd or Thursday April 3rd
- **Last time**
  - Chapel introduction
  - Please start the docker pull for Chapel (see next slide)
- **Today**
  - TopHat questions about ChapelCon tutorial and last week's class, aka ICA10 prep
  - Kmer counting example: file IO, maps, strings
  - Parallel processing of files
  - Overview of GPU programming support

HANDS ON: HOW TO DO THE HANDS ON

01-hello.chpl

Example codes for Chapel tutorial slides

- https://github.com/UofA-CSc-372-Spring-2025/CSc372Spring2025-CourseMaterials/tree/main/Sandboxes/ChapelTutorialExamples

Using a container on your laptop

- First, install docker for your machine and start it up (see the README.md for more info)
- Then, use the chapel-gasnet docker container

```
docker pull docker.io/chapel/chapel-gasnet     # takes about 5 minutes
cd CSc372Spring2025-CourseMaterials/Sandboxes/ChapelTutorial/
docker run --rm -it -v "$PWD":/workspace chapel/chapel-gasnet
root@589405d07f6a:/opt/chapel# cd /workspace
root@xxxxxxxxx:/myapp# chpl 01-hello.chpl
root@xxxxxxxxx:/myapp# ./01-hello -nl 1
```

HANDS ON: PARALLELISM AND LOCALITY IN CHAPEL

Goals

- Experiment some with '01-basics-distarr.chpl'

```
chpl 01-basics-distarrchpl
./01-basics-distarr -nl 1
./01-basics-distarr -nl 4
```

Experiment some with '01-basics-distarr.chpl'

1. What happens when you add a 'writeln(D)' to write out the domain 'D'?
2. What happens when you change 'D's initial value to '{0..3,0..3}'?
3. Use a config const for the upper bound of the domain 'D'.
4. Have array A and array B use the same domain.

OUTLINE: OVERVIEW OF PROGRAMMING IN CHAPEL

- Chapel Goals, Usage, and Comparison with other Tools
- Hello World (Hands On)
- Chapel Execution Model and Parallel Hello World (Hands On)
- kmer counting using file IO, config consts, strings, maps (Hands On)
- Parallelizing a program that processes files (Hands On)
- GPU programming support
- Learning goals for rest of Chapel unit

KMER COUNTING USING FILE IO, CONFIG CONSTS, AND STRINGS (HANDS ON)

SERIAL CODE USING MAP/DICTIONARY: K-MER COUNTING

kmer.chpl

kmer.chpl

```
use Map, IO;

config const infilename = "kmer_large_input.txt";
config const k = 4;

var sequence, line : string;
var f = open(infilename, ioMode.r);
var infile = f.reader();
while infile.readLine(line) {
 sequence += line.strip();
}

var nkmerCounts : map(string, int);

for ind in 0..<(sequence.size-k) {
 nkmerCounts[sequence[ind..#k]] += 1;
}
```

‘Map’ and 'IO' are two of the standard libraries provided in Chapel. A 'map' is like a dictionary in python.

'config const' indicates a configuration constant, which result in built-in command-line parsing

Reading all of the lines from the input file into the string 'sequence'.

The variable 'nkmerCounts' is being declared as a dictionary mapping strings to ints

Counting up each kmer in the sequence

HANDS ON: EXPERIMENTING WITH THE K-MER EXAMPLE

kmer.chpl

Some things to try out with 'kmer.chpl'

```
chpl kmer.chpl
./kmer -nl 1
./kmer -nl 1 --k=10                              # can change k
./kmer -nl 1 --infilename="kmer.chpl"            # changing infilename
./kmer -nl 1 --k=10 --infilename="kmer.chpl" # can change both
```

Experiment some with kmer.chpl

1. When k=5, what is the most common kmer in the file from doing "wget https://www.bioinformatics.nl/tools/crab_fasta.html"
2. When k=8?

Key concepts

- 'use' command for including modules
- configuration constants, 'config const'
- reading from a file
- 'map' data structure

PARALLELIZING A PROGRAM THAT PROCESSES FILES (HANDS ON)

ANALYZING MULTIPLE FILES USING PARALLELISM

parfilekmer.chpl

parfilekmer.chpl

```
use FileSystem, BlockDist;
config const dir = "DataDir";
var fList = findFiles(dir);
var filenames =
  blockDist.createArray(0..<fList.size,string);
filenames = fList;

// per file word count
forall f in filenames {
   ...
   // code from kmer.chpl
   ...
}
```

```
prompt> chpl --fast parfilekmer.chpl
prompt> ./parfilekmer -nl 1
prompt> ./parfilekmer -nl 4
```

- shared and distributed-memory parallelism using 'forall'
  - in other words, parallelism within the locale/node and across locales/nodes
- a distributed array
- command line options to indicate number of locales

Experiment some with parfilekmer.chpl

1. Using ‘writeln’, edit parfilekmer.chpl so that ‘fList’ is printed to the screen.
2. Now print out the ‘filenames’ array.

BLOCK DISTRIBUTION OF ARRAY OF STRINGS

Locale 0 **Locale 1**

"filename1"	"filename2"	"filename3"	"filename4"	"filename5"	"filename6"	"filename7"	"filename8"

- Array of strings for filenames is distributed across locales

- 'forall' will do parallelism across locales and then within each locale to take advantage of multicore

HANDS ON: PROCESSING FILES IN PARALLEL

parfilekmer.chpl

Some things to try out with 'parfilekmer.chpl'

```
chpl parfilekmer.chpl --fast
./parfilekmer -nl 2 --dir="SomethingElse/"     # change dir with inputs files

./parfilekmer -nl 2 --k=10                     # can also change k
```

Concepts illustrated

- 'forall' provides distributed and shared memory parallelism when do a 'forall' over the Block distributed array
- No remote puts and gets

GPU PROGRAMMING SUPPORT

GPU SUPPORT IN CHAPEL

Generate code for GPUs

- Support for NVIDIA and AMD GPUs
- Exploring Intel support

Key concepts

- Using the 'locale' concept to indicate execution and data allocation on GPUs
- 'forall' and 'foreach' loops are converted to kernels
- Arrays declared within GPU sublocale code blocks are allocated on the GPU

Chapel code calling CUDA examples

- https://github.com/chapel-lang/chapel/blob/main/test/gpu/interop/stream/streamChpl.chpl
- https://github.com/chapel-lang/chapel/blob/main/test/gpu/interop/cuBLAS/cuBLAS.chpl

For more info...

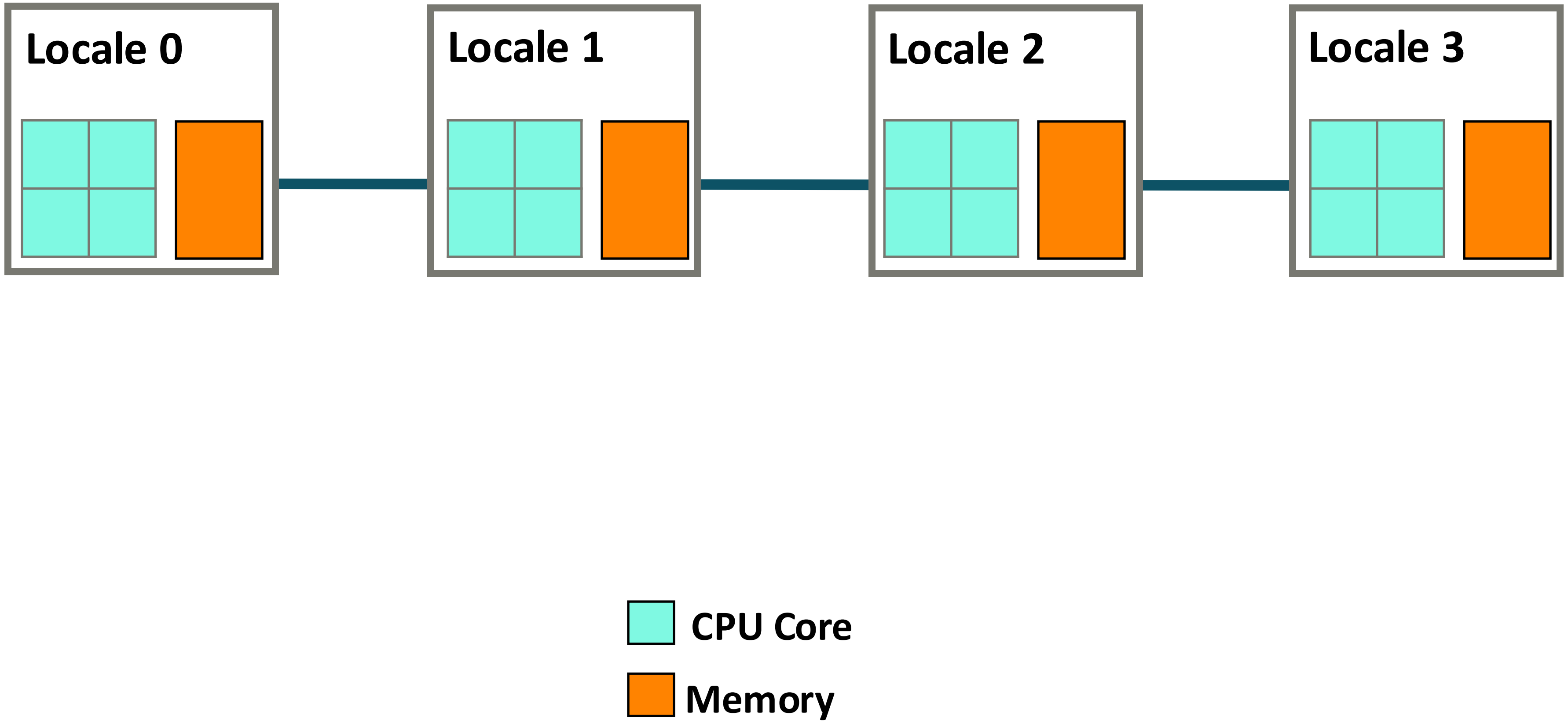
– https://chapel-lang.org/docs/technotes/gpu.html

gpuExample.chp

```
use GpuDiagnostics;
startGpuDiagnostics();

var operateOn =
if here.gpus.size>0 then here.gpus
                    else [here,];

// Same code can run on GPU or CPU
coforall loc in operateOn do on loc {
 var A : [1..10] int;
 foreach a in A do a+=1;
  writeln(A);
}

stopGpuDiagnostics();
writeln(getGpuDiagnostics());
```

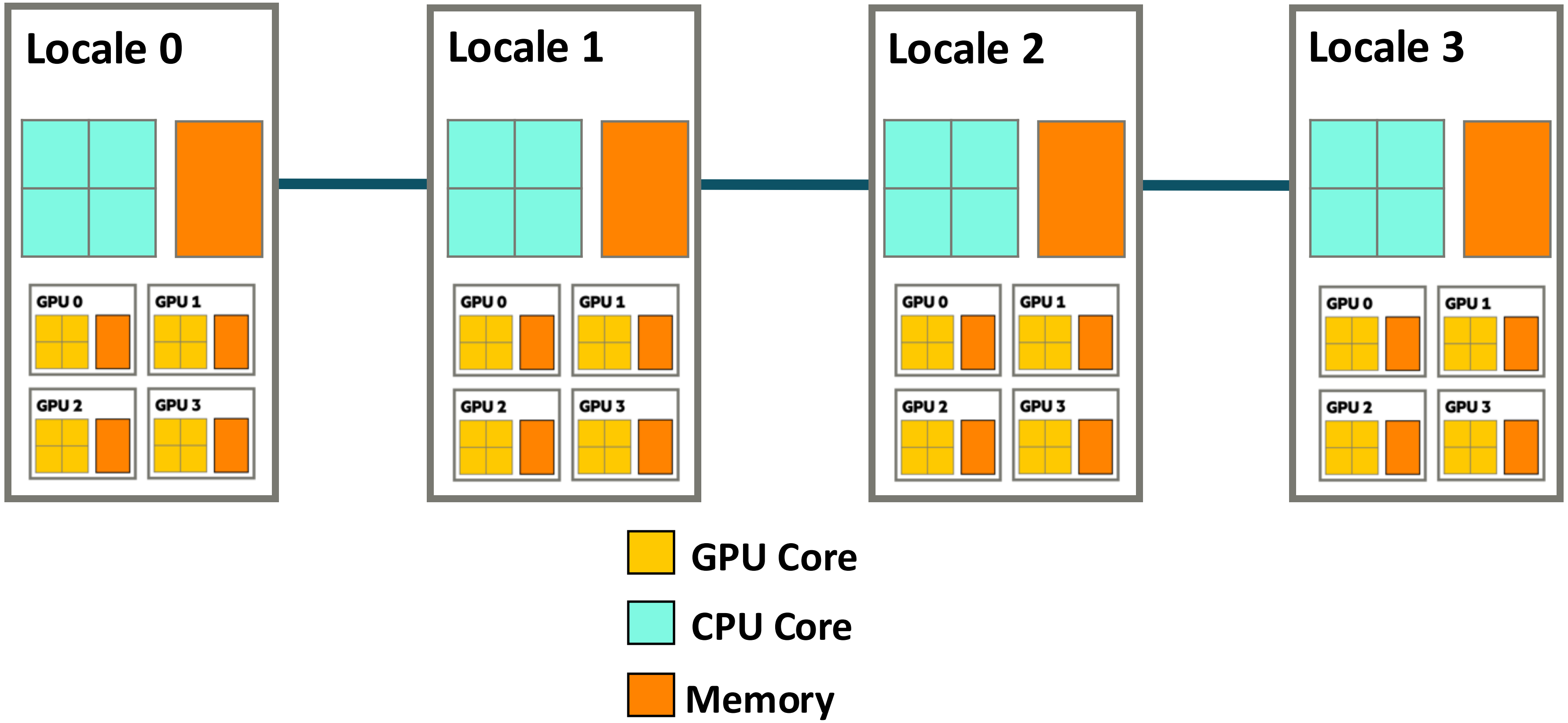
KEY CONCERNS FOR SCALABLE PARALLEL COMPUTING

1. **parallelism:** Which tasks should run simultaneously?
2. **locality:** Where should tasks run? Where should data be allocated?
   - complicating matters, compute nodes now often have GPUs with their own processors and memory

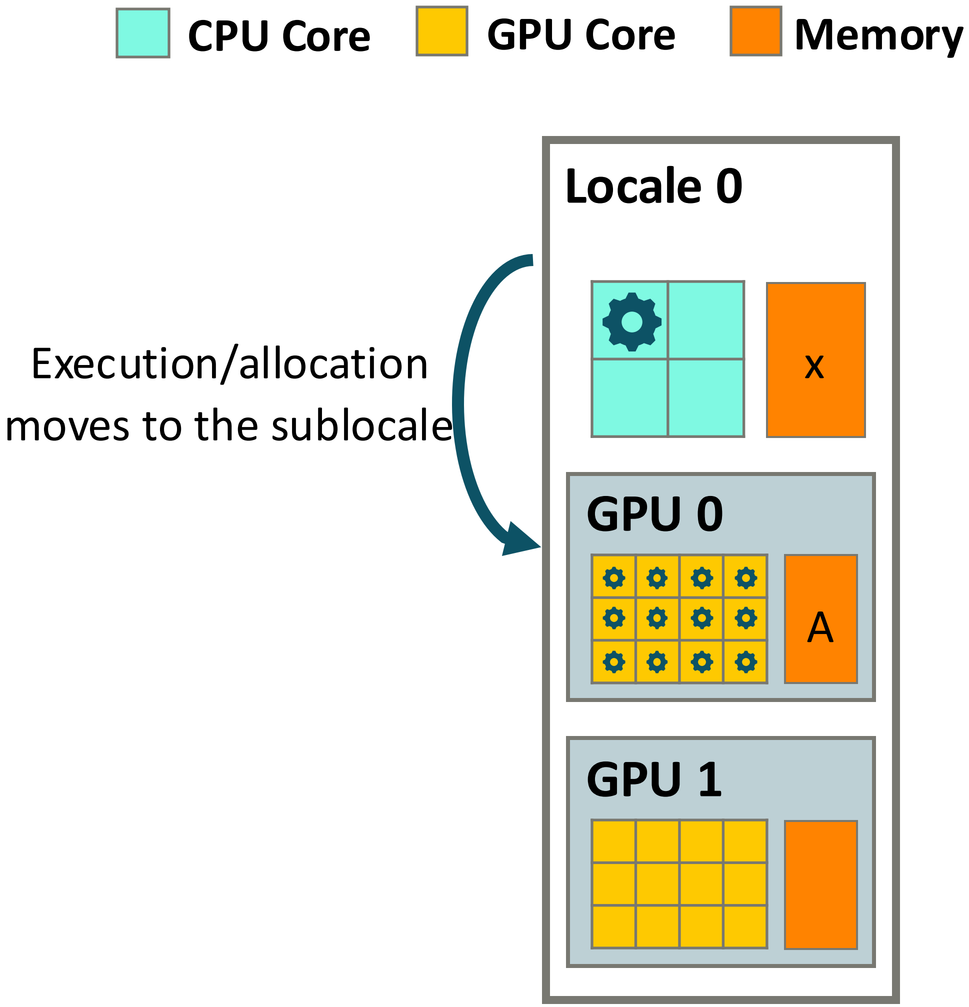


KEY CONCERNS FOR SCALABLE PARALLEL COMPUTING

1. **parallelism:** Which tasks should run simultaneously?
2. **locality:** Where should tasks run? Where should data be allocated?
   - complicating matters, compute nodes now often have GPUs with their own processors and memory
   - we represent these as *sub-locales* in Chapel

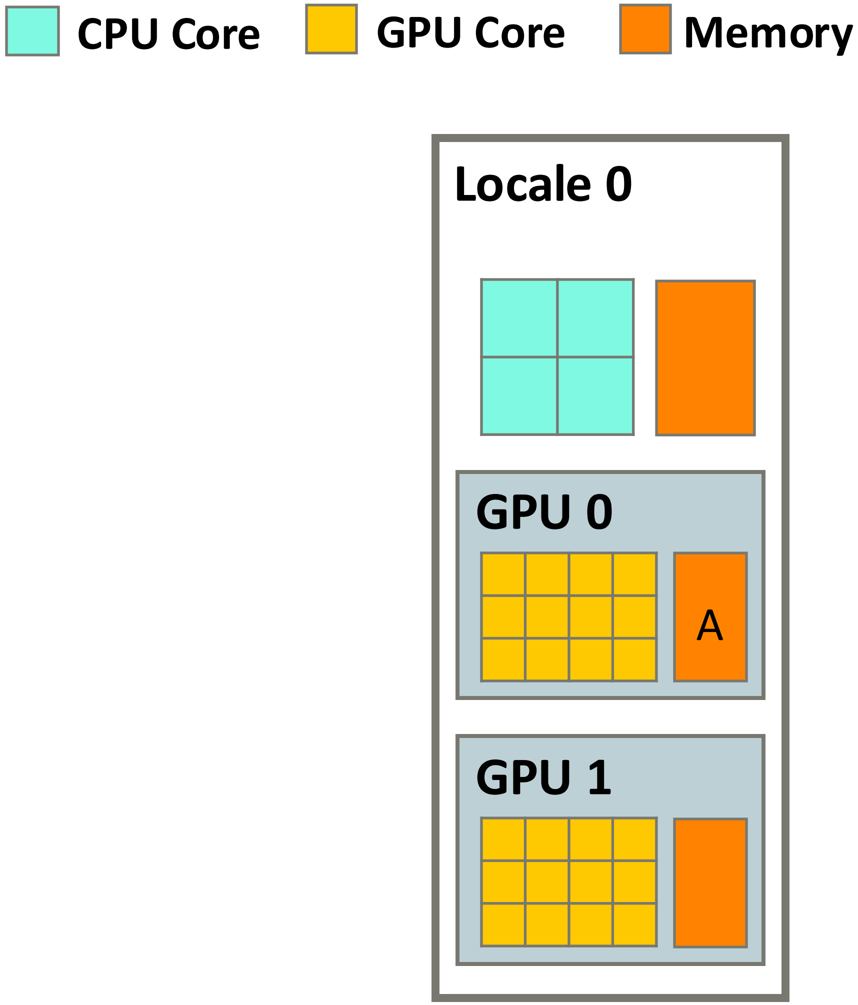


PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS



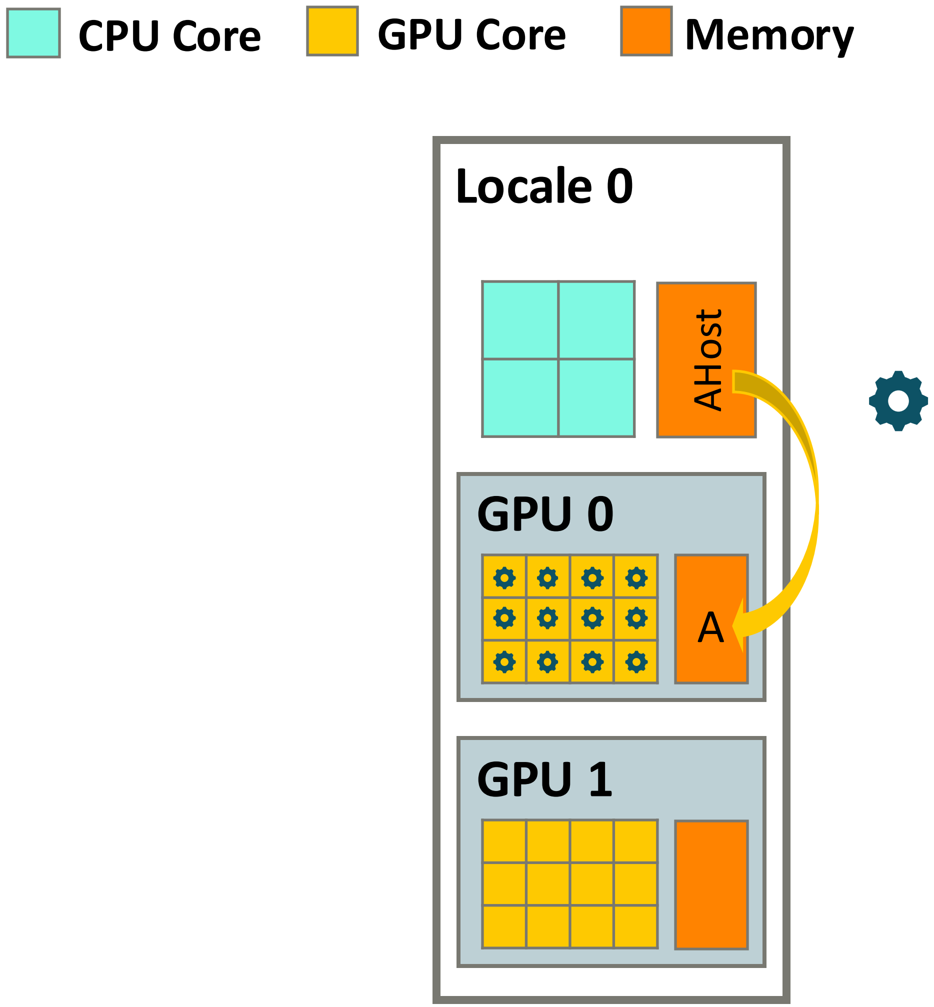

```
var x = 10;

  on here.gpus[0] {
   var A = [1, 2, 3, 4, 5, ...];
   foreach a in A do a += 1;
  }

writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS



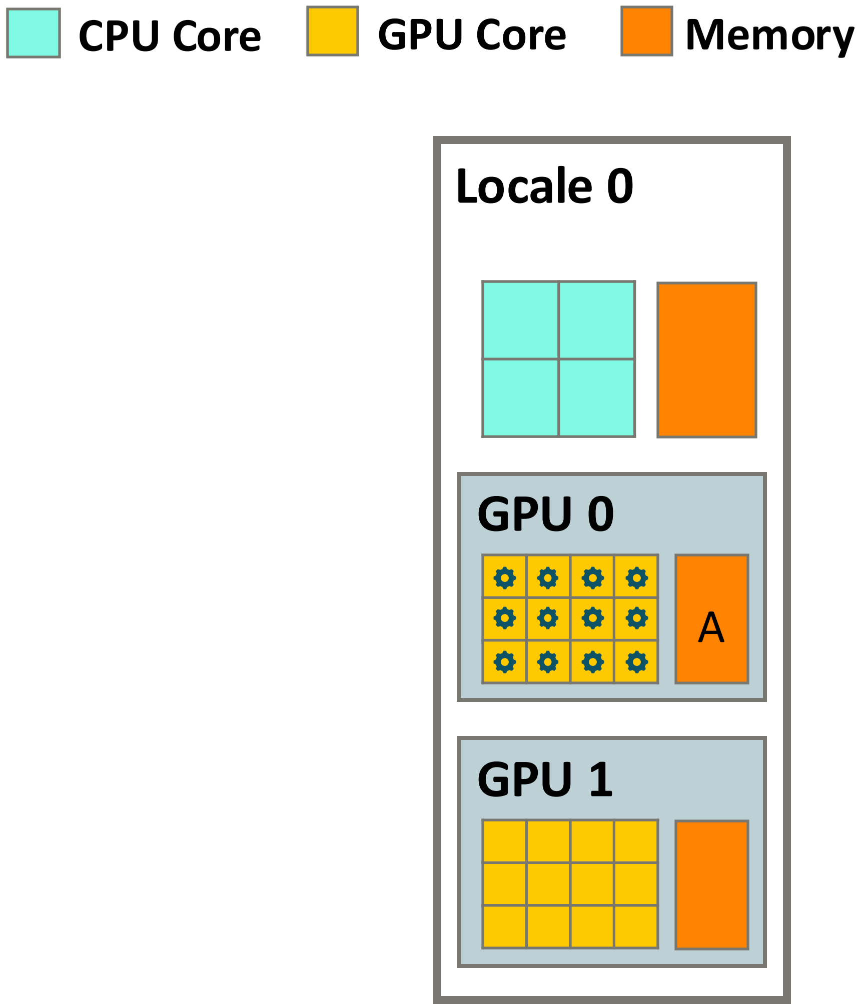

```
var x = 10;


  on here.gpus[0] {
   var A = [1, 2, 3, 4, 5, ...];
   foreach a in A do a += 1;
  }






writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS



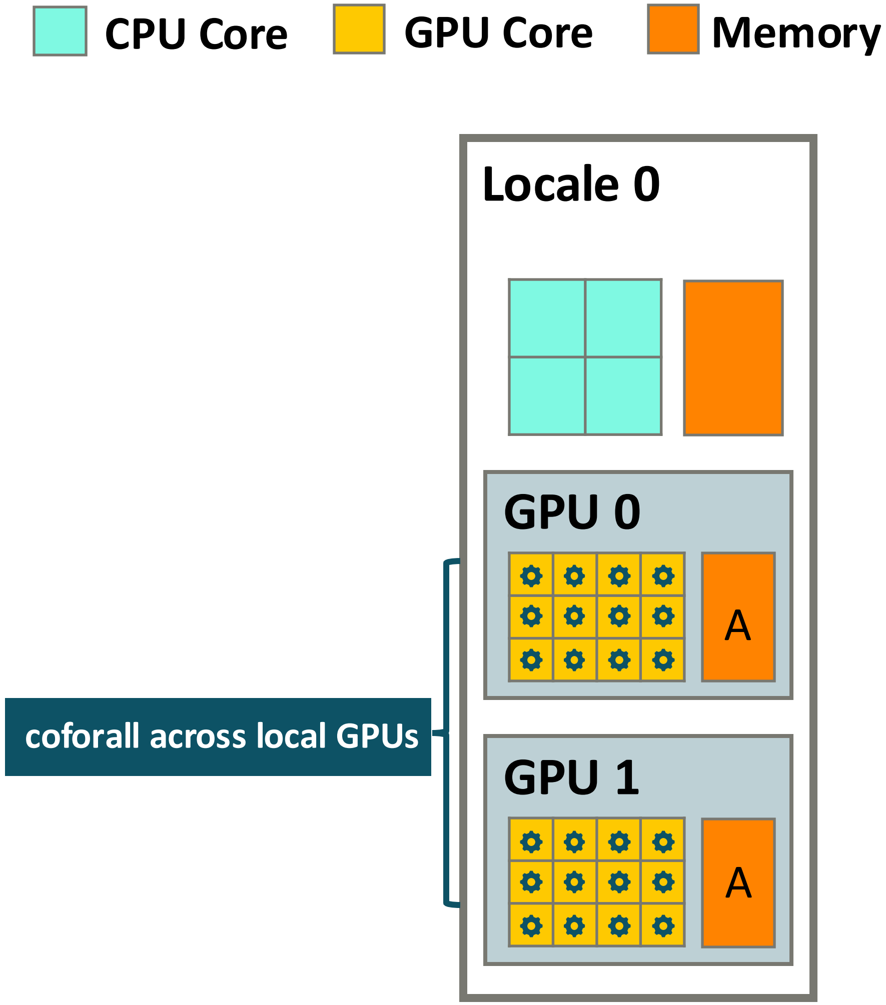

```
var x = 10;
var AHost = [1, 2, 3, 4, 5, ...];

  on here.gpus[0] {
   var A = AHost;
   foreach a in A do a += 1;
  }


writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS



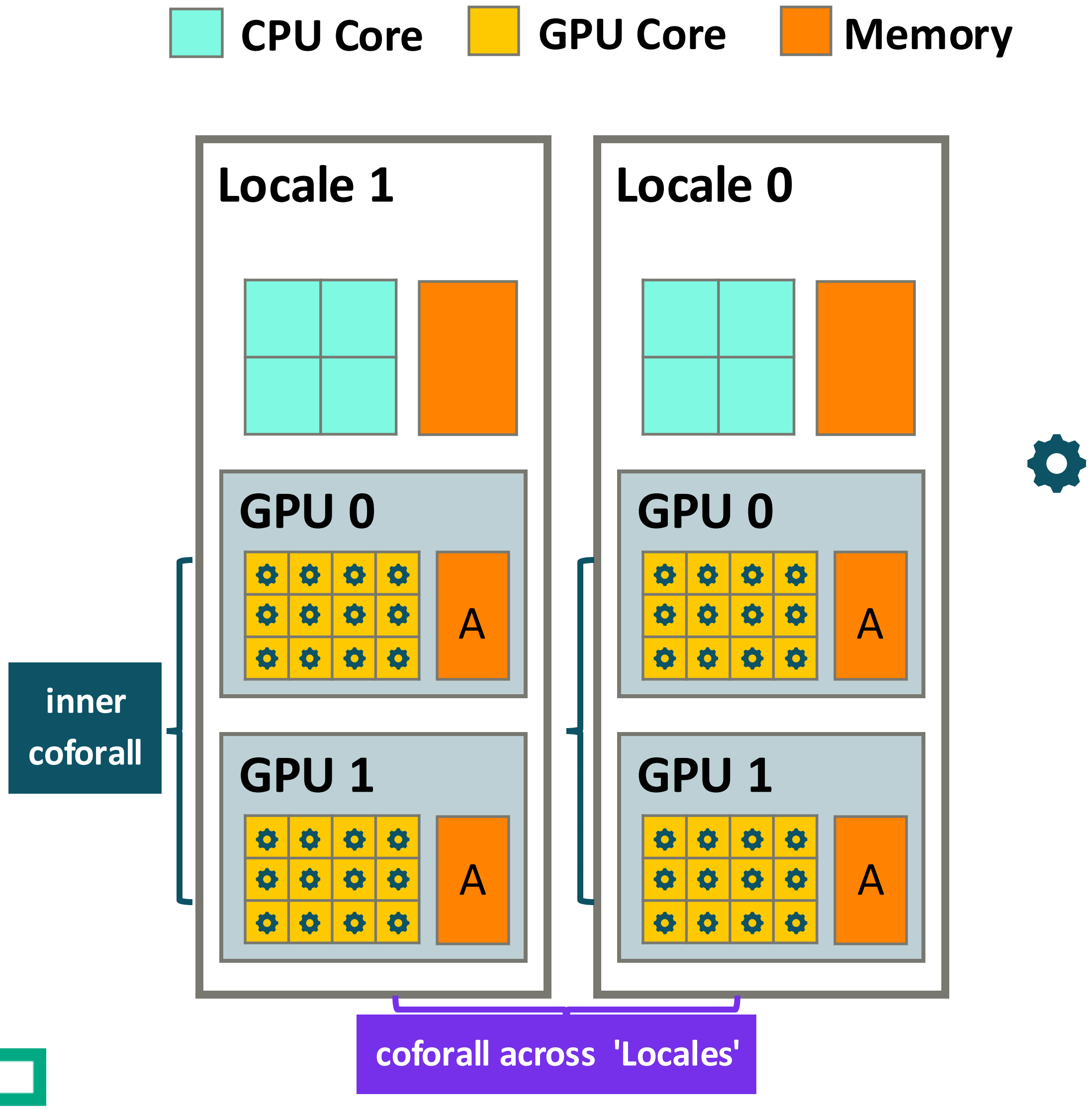

```
var x = 10;


  on here.gpus[0] {
    var A = [1, 2, 3, 4, 5, ...];
    foreach a in A do a += 1;
  }





writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS



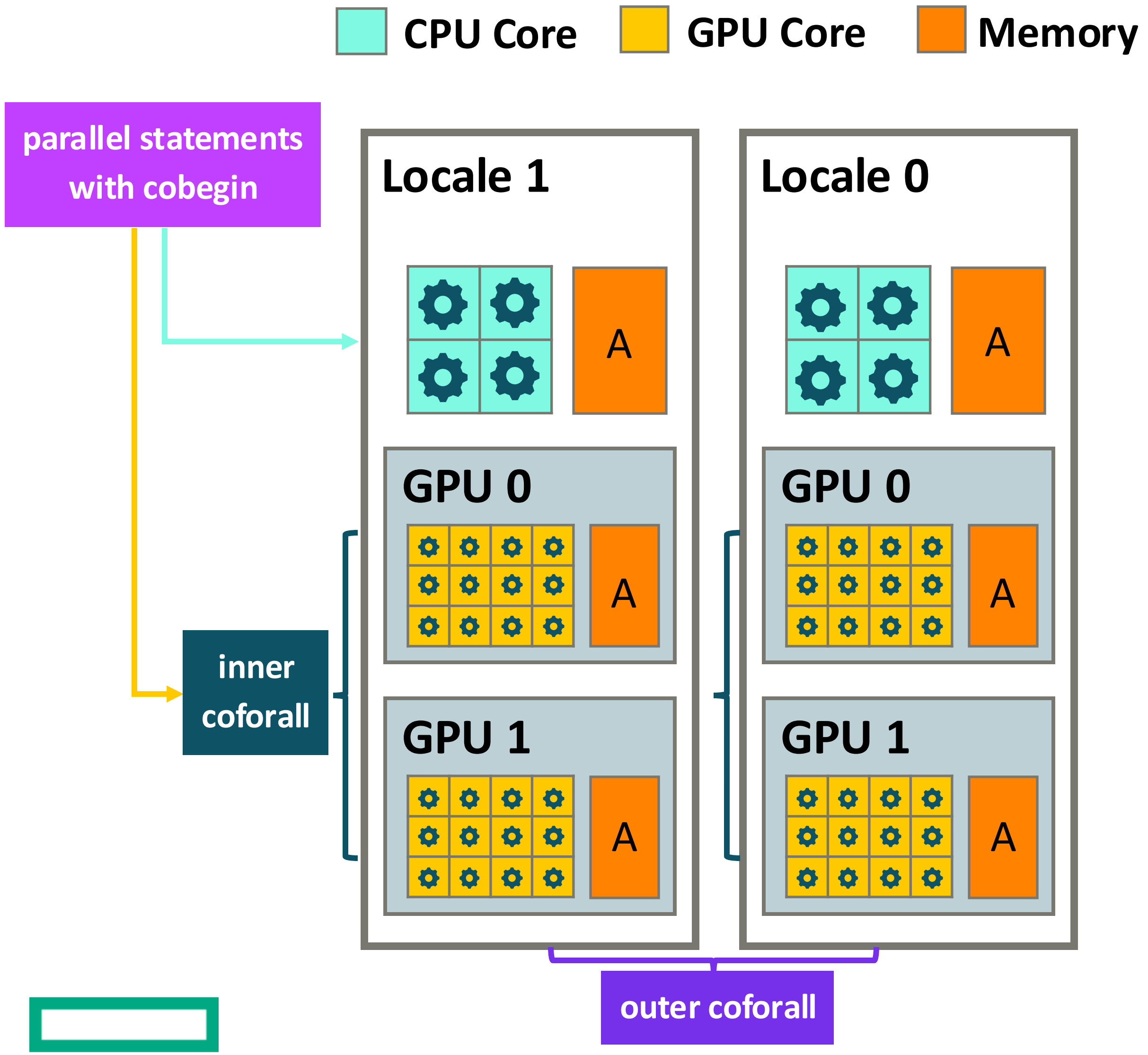

```
var x = 10;

coforall g in here.gpus do on g {
  var A = [1, 2, 3, 4, 5, ...];
  foreach a in A do a += 1;
}

writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS




```
var x = 10;
coforall l in Locales do on l {

  coforall g in here.gpus do on g {
   var A = [1, 2, 3, 4, 5, ...];
   foreach a in A do a += 1;
  }

}
writeln(x);
```

PARALLELISM AND LOCALITY IN THE CONTEXT OF GPUS




```
var x = 10;
coforall l in Locales do on l {
  cobegin {
    coforall g in here.gpus do on g {
      var A = [1, 2, 3, 4, 5, ...];
      foreach a in A do a += 1;
    }
    {
      var A = [1, 2, 3, 4, 5, ...];
      foreach a in A do a += 1;
    }
  }
}
writeln(x);
```

LEARNING OBJECTIVES FOR THE REST OF THE CHAPEL UNIT

LEARNING OBJECTIVES FOR CHAPEL UNIT

- Familiarity with the Chapel execution model including how to run codes in parallel on a single node, across nodes, and both
- Learn Chapel concepts by compiling and running provided code examples
  - ✓Serial code using map/dictionary, (k-mer counting from bioinformatics)
  - ✓Parallelism and locality in Chapel
  - ✓Distributed parallelism and 1D arrays, (processing files in parallel)
  - Chapel basics in the context of an n-body code
  - Distributed parallelism and 2D arrays, (heat diffusion problem)
  - How to parallelize histogram
  - Using CommDiagnostics for counting remote reads and writes
  - Chapel and Arkouda best practices including avoiding races and performance gotchas
- Where to get help and how you can participate in the Chapel community
- Memory safety in Chapel and other languages like Rust

OTHER CHAPEL EXAMPLES & PRESENTATIONS

Primers

- https://chapel-lang.org/docs/primers/index.html

Blog posts for Advent of Code

- https://chapel-lang.org/blog/index.html

Examples people have written

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

Test directory in main repository

- https://github.com/chapel-lang/chapel/tree/main/test

Presentations

- https://chapel-lang.org/presentations.html

CHAPEL RESOURCES

Chapel homepage: https://chapel-lang.org
- (points to all other resources)

Social Media:
- Twitter: @ChapelLanguage
- Facebook: @ChapelLanguage
- YouTube: http://www.youtube.com/c/ChapelParallelProgrammingLanguage

Community Discussion / Support:
- Discord: https://discord.com/invite/xu2xg45yqH
- Stack Overflow: https://stackoverflow.com/questions/tagged/chapel
- GitHub Issues: https://github.com/chapel-lang/chapel/issues