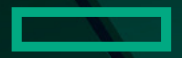


Hewlett Packard
Enterprise

CHAPEL TUTORIAL FOR PYTHON PROGRAMMERS: PRODUCTIVITY AND PERFORMANCE IN ONE LANGUAGE

Michelle Strout, Jeremiah Corrado, and Scott Bachman
University of Arizona
February 3, 2023

HOW TO PARTICIPATE IN THIS TUTORIAL

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **Poll Everywhere link**: pollev.com/michellestrout402
  - There will be fun questions throughout the tutorial

- **Attempt this Online website for running Chapel code**
  - Go to main Chapel webpage at https://chapel-lang.org/
  - Click on the little ATO icon on the lower left that is above the YouTube icon

- **Using a container on your laptop**
  - First, install podman or docker for your machine and then start them up
  - Then, the below commands work with podman or docker

```
podman pull docker.io/chapel/chapel     # takes about 3 minutes
cd ChapelForPythonProgrammersFeb2023    # assuming git clone has happened
podman run --rm -v "$PWD":/myapp -w /myapp chapel/chapel chpl hello.chpl
podman run --rm -v "$PWD":/myapp -w /myapp chapel/chapel ./hello
```

- **Chapel on Puma and Ocelote:** see the README.md in repository

CHAPEL PROGRAMMING LANGUAGE

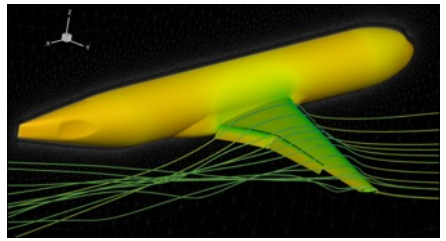
Chapel is a general-purpose programming language that provides

ease of parallel programming,
high performance, and
portability.

And is being used in applications in various ways:

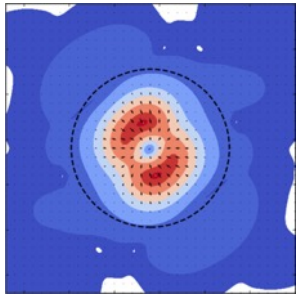
refactoring existing codes,
developing new codes,
serving high performance to Python codes **(Chapel server with Python client),** and
providing distributed and shared memory parallelism for existing codes.

HOW APPLICATIONS ARE USING CHAPEL



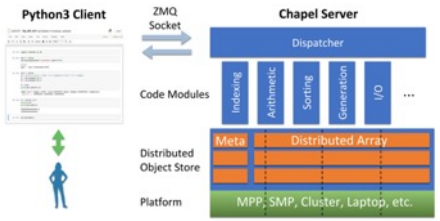
Refactoring existing codes into Chapel (~100K lines of Chapel)

CHAMPS: 3D Unstructured CFD
Éric Laurendeau, Simon Bourgault-Côté, Matthieu Parenteau, et al.
École Polytechnique Montréal



Writing code in Chapel
(~10k lines of including parallel FFT)

ChplUltra: Simulating Ultralight Dark Matter
Nikhil Padmanabhan, J. Luna Zagorac, *et al.*
Yale University / University of Auckland



Chapel server for a Python client (~25K lines of Chapel)

Arkouda: NumPy at Massive Scale
Mike Merrill, Bill Reus, et al.
US DoD

INTRODUCTIONS

- Let's take some time to briefly introduce ourselves
  - Michelle
    - Chapel team leader
    - Affiliate faculty in the Department of Computer Science
  - Current Chapel team
  - Team members on slack
    - Jeremiah Corrado
    - Daniel Fedorin
  - **Participants, tell us some about yourself**

LEARNING OBJECTIVES FOR TODAY'S TUTORIAL

- Compile and run Chapel programs in a web browser, on Puma/Ocelote, and/or on your laptop
- Familiarity with the Chapel execution model including how to run codes in parallel on a single node, across nodes, and both
- Experiment compiling and running provided Chapel code examples
  - k-mer counting (bioinformatics application)
  - Processing files in parallel using parallelism over multiple nodes and threads
  - Solving a diffusion PDE (partial differential equation)
  - Image processing (coral reef diversity example)
  - Same code can be compiled to run on a multi-core CPU AND a GPU
- Where to get help and how you can participate in the Chapel community

HOW TO PARTICIPATE IN THIS TUTORIAL

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **Attempt this Online website for running Chapel code**
  - Go to main Chapel webpage at https://chapel-lang.org/
  - Click on the little ATO icon on the lower left that is above the YouTube icon

- **Using a container on your laptop**
  - First, install podman or docker for your machine and then start them up
  - Then, the below commands work with podman or docker

```
podman pull docker.io/chapel/chapel     # takes about 3 minutes
cd ChapelForPythonProgrammersFeb2023    # assuming git clone has happened
podman run --rm -v "$PWD":/myapp -w /myapp chapel/chapel chpl hello.chpl
podman run --rm -v "$PWD":/myapp -w /myapp chapel/chapel ./hello
```

- **Chapel on Puma and Ocelote:** see the README.md in repository

Try one of these three options for using Chapel

Which option did you choose to try out Chapel during this tutorial?

Attempt This Online

Container on your laptop

Puma/Ocelote

PARALLELISM ACROSS NODES AND WITHIN NODES

- **Parallel hello world**
  - hellopar.chpl
- **Key concepts**
  - 'coforall'
  - configuration constants, 'config const'
  - range values, '0..#tasksPerLocale'
    - potentially via separate compilation / incremental recompilation
  - 'writeln'
  - inline comments start with '//'

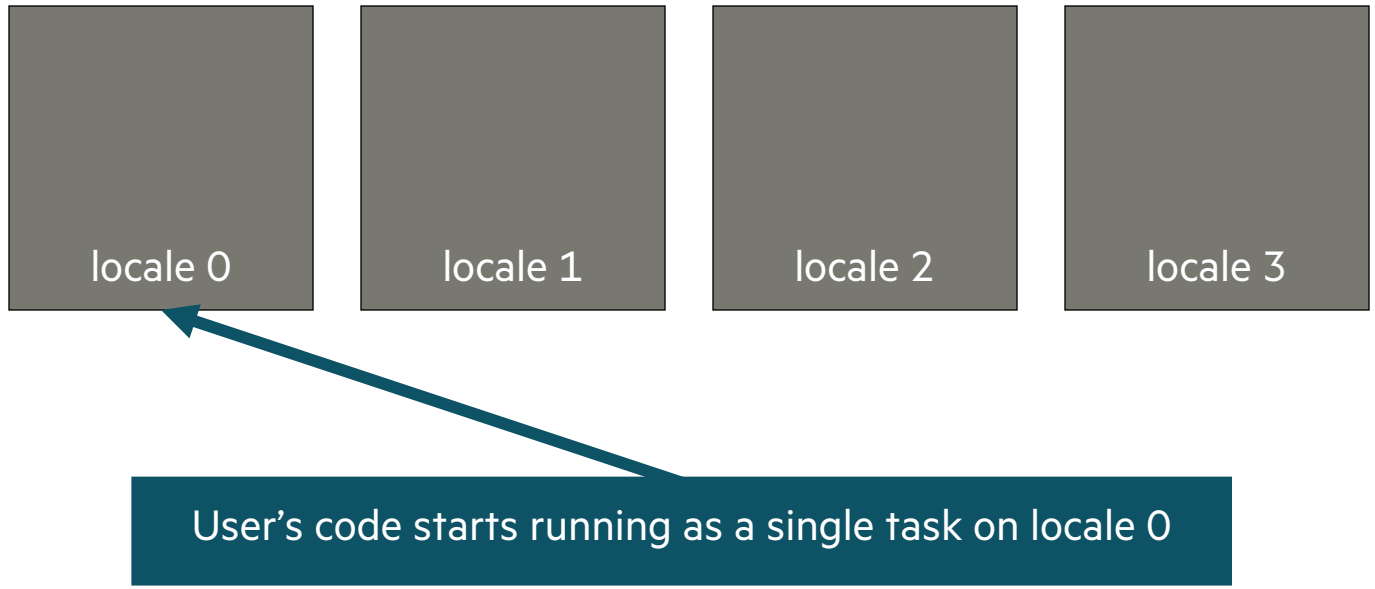
```
// can be set on the command line with --tasksPerLocale=2
config const tasksPerLocale = 1;

// parallel loops over nodes and then over threads
coforall loc in Locales on loc {
  coforall tid in 0..#tasksPerLocale {

    writeln("Hello world! ",
            "(from task ", tid,
            " of ", tasksPerLocale,
            " on locale ", here.id,
            " of ", numLocales, ")" );
  }
}
```

CHAPEL EXECUTION MODEL AND TERMINOLOGY: LOCALES

- Locales can run tasks and store variables
  - Think “compute node” on a parallel system
  - User specifies number of locales on executable’s command-line

```
prompt> ./myChapelProgram --numLocales=4    # or ‘-nl 4’
```

`Locales` array:



TASK-PARALLEL “HELLO WORLD”

helloTaskPar.chpl

```
const numTasks = here.numPUs();
coforall tid in 1..numTasks do
  writef("Hello from task %n of %n on %s\n",
         tid, numTasks, here.name);
```

TASK-PARALLEL "HELLO WORLD"

helloTaskPar.chpl

```
const numTasks = here.numPUs();
coforall tid in 1..numTasks do
  writef("Hello from task %n of %n on %s\n",
         tid, numTasks, here.name);
```

'here' refers to the locale on which we're currently running

how many processing units (think "cores") does my locale have?

what's my locale's name?

TASK-PARALLEL “HELLO WORLD”

helloTaskPar.chpl

```
const numTasks = here.numPUs();
coforall tid in 1..numTasks do
  writef("Hello from task %n of %n on %s\n",
         tid, numTasks, here.name);
```

a 'coforall’ loop executes each iteration as an independent task

```
prompt> chpl helloTaskPar.chpl
prompt> ./helloTaskPar
Hello from task 1 of 4 on n1032
Hello from task 4 of 4 on n1032
Hello from task 3 of 4 on n1032
Hello from task 2 of 4 on n1032
```

TASK-PARALLEL "HELLO WORLD"

helloTaskPar.chpl

```
const numTasks = here.numPUs();
coforall tid in 1..numTasks do
  writef("Hello from task %n of %n on %s\n",
         tid, numTasks, here.name);
```

```
prompt> chpl helloTaskPar.chpl
prompt> ./helloTaskPar
Hello from task 1 of 4 on n1032
Hello from task 4 of 4 on n1032
Hello from task 3 of 4 on n1032
Hello from task 2 of 4 on n1032
```

So far, this is a shared-memory program

Nothing refers to remote locales, explicitly or implicitly

TASK-PARALLEL "HELLO WORLD" (DISTRIBUTED VERSION)

helloTaskPar.chpl

```
coforall loc in Locales {
  on loc {
    const numTasks = here.numPUs();
    coforall tid in 1..numTasks do
      writef("Hello from task %n of %n on %s\n",
             tid, numTasks, here.name);
  }
}
```

TASK-PARALLEL "HELLO WORLD" (DISTRIBUTED VERSION)

helloTaskPar.chpl

```
coforall loc in Locales {
  on loc {
    const numTasks = here.numPUs();
    coforall tid in 1..numTasks do
      writef("Hello from task %n of %n on %s\n",
             tid, numTasks, here.name);
  }
}
```

create a task per locale on which the program is running

have each task run 'on' its locale

then print a message per core, as before

```
prompt> chpl helloTaskPar.chpl
prompt> ./helloTaskPar -nl=4
Hello from task 1 of 4 on n1032
Hello from task 4 of 4 on n1032
Hello from task 1 of 4 on n1034
Hello from task 2 of 4 on n1032
Hello from task 1 of 4 on n1033
Hello from task 3 of 4 on n1034
Hello from task 1 of 4 on n1035
…
```

When poll is active, respond at **pollev.com/michellestrout402**

Which Chapel code does the same thing as this python code?

```
x = 42
str = "answer"
print(str, " = ", x)
```

A

B

C

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

A

```
var x = 42;
var str = "answer";
writeln(str, " = ", x);
```

B

```
config const tasksPerLocale = 2;
coforall tid in 0..#tasksPerLocale {
  var message = "answer = ";
  message += 42:string;
  writeln(message);
}
```

C

```
var x = 42;
var str = "answer";
coforall loc in Locales {
  on loc {
    writeln(x, " = ", str);
  }
}
```

K-MER COUNTING FROM BIOINFORMATICS

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();
}
infile.close();

var nkmerCounts : map(string, int);

for in for 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

EXPERIMENTING WITH THE K-MER EXAMPLE

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **Some things to try out with 'kmer.html'**

```
chpl kmer.html
./kmer

./kmer --k=10                                  # can change k
./kmer --infilename="kmer.chpl"                # can change the infilename
./kmer --k=10 --infilename="kmer.chpl"         # can change both
```

When poll is active, respond at **pollev.com/michellestrout402**

What Chapel code does the same thing as this python code?

```python
# read in a file into a list of strings
# where each string has a line with the newline at the end removed
with open("filename.txt") as file:
  lines = [line.strip() for line in file]

print(lines)
```

A

B

C

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

A

```
// declare a dictionary/map to store the count per kmer
var nkmerCounts : map(string, int);

// count up the number of times each kmer occurs
for ind in 0..<(sequence.size-k) {
  nkmerCounts[sequence[ind..#k]] += 1;
}
```

B

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

C

```
use List, IO;

var line : string;
var lines : list(string);
var infile = open("filename.txt",iomode.r).reader();
while infile.readLine(line) {
  lines.append(line.strip());
}

writeln(lines);
```

POLL EV: PICK AN ENGLISH DESCRIPTION

- string usage, after kmer example
  - parse a string into an integer and vice versa
  - count characters in a string
  - concatenate two strings
  - find a substring
  - split on colon

2D DIFFUSION PARTIAL DIFFERENTIAL EQUATION EXAMPLE

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **See 'diffusion.chpl' in the repository**
- **Some things to try out with 'diffusion.html'**

```
chpl diffusion.html
./diffusion

--xLen=4 --yLen=4 --nx=61 --ny=61   # doubles the size of the domain along each
                                    # dimension, keeping the density of points the same

--nu=0.025                          # reduces the viscosity of the fluid

--nt=100                            # runs the simulation for twice as long
```

POLL EV: MORE REDUCE EXAMPLES

- Reduces work on arrays, lists, sets, FIXME: is there a list somewhere? Pick which Chapel code snippet reads in a list of integers from a file and then computes the minimal value?
  - Have like four code examples showing different reduces working on different kinds of data structures

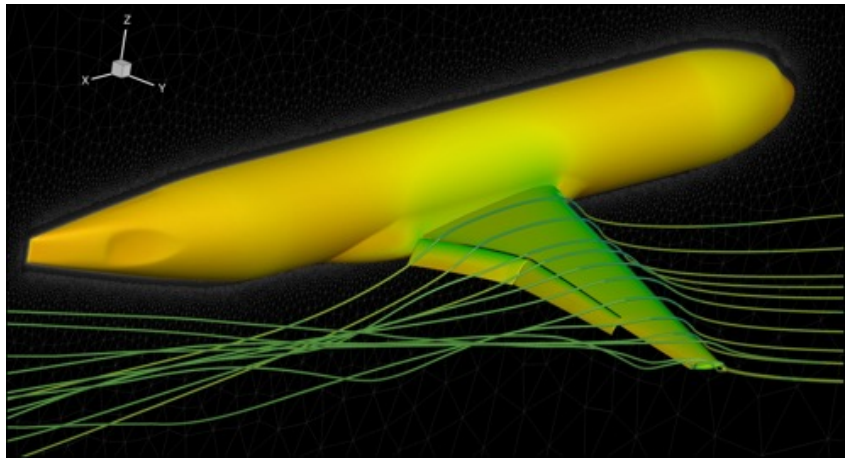
CHAMPS IN ONE SLIDE

What is it?

- Computational Fluid Dynamics framework for airplane simulation written from scratch
- Modular design, permitting various computational modules to be integrated (or not)
- ~48k lines written in ~2 years

Who did it?

- Professor Eric Laurendeau's team at Polytechnique Montreal
- not open-source (yet), but available by request to researchers



Why Chapel?

- performance and scalability competitive with MPI + C++
- provided a simpler coding experience for computational scientists
  - has enabled senior students to move faster
  - has permitted junior students to contribute more readily
- net result: achieves competitive results w.r.t. established, world-class frameworks from Stanford, MIT, etc.

CHAMPS: QUOTES AND STATUS FROM THE PI

- Eric Laurendeau (PI) gave our CHIUW 2021 keynote
  - title: *HPC Lessons From 30 Years of Practice in CFD Towards Aircraft Design and Analysis*
  - students also gave talks on their individual efforts
  - key excerpts:

    "To show you what Chapel did in our lab… [NSCODE, their previous framework] ended up 120k lines. And my students said 'We can't handle it anymore. It's too complex, we lost track of everything.' And today, they went from 120k lines to 48k lines, so 3x less. But the code is not 2D, it's 3D. And it's not structured, it's unstructured, which is way more complex. And it's multi-physics: aeroelastic, aeroicing. So, I've got industrial-type code in 48k lines. So for me, this is like the proof of the benefit of Chapel, plus the smiles I have on my students everyday in the lab because they love Chapel as well. So that's the key, that's the takeaway."

    "So CHAMPS, that's the new solver that has been made, and all made by the students… So, [Chapel] promotes the programming efficiency. It was easy for them to learn. …I see the end result. We ask students at the master's degree to do stuff that would take 2 years and they do it in 3 months. And I'm not joking, this is from 2 years to 3 months. So if you want to take a summer internship and you say 'program a new turbulance model', well they manage. And before, it was impossible to do."
- CHAMPS participating in 4th CFD High Lift Prediction Workshop and 1st Icing Prediction Workshop
  - teams compete against one another to do the same massive simulations
    - entries compared in terms of model accuracy, performance, practicality
  - sponsored by AIAA and NASA
  - initial results are looking competitive to longer-lived / more established codes from Stanford, MIT, etc.

WRITING OUT EVERYTHING EXAMPLE

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **See 'diffusion.chpl' in the repository**
- **Some things to try out with 'diffusion.html'**

```
chpl kmer.html
./kmer

./kmer --k=10                                  # can change k
./kmer --infilename="kmer.chpl"                # can change the infilename
./kmer --k=10 --infilename="kmer.chpl"         # can change both
```

POLL EV: MAP/DICTIONARY USAGE EXAMPLES

- dictionary usage
  - iterate over pairs, keys, or values (is this potentially going to change?)

```
dt = {'a': 'juice', 'b': 'grill', 'c': 'corn'}

for key, value in dt.items():
print(key, value)
```

  - sort the keys in a dictionary
  - histogram example

POLL EV: SET USAGE EXAMPLES

- all unique elements in a list,

```
print(list(set(list_1)))
```

```
# Program to perform different set operations like in mathematics

# define three sets
E = {0, 2, 4, 6, 8};
N = {1, 2, 3, 4, 5};

# set union
print("Union of E and N is",E | N)

# set intersection
```

ANALYZING MULTIPLE FILES USING PARALLELISM

parfilekmer.chpl

```
use FileSystem;
config const dir = "DataDir";
var fList = findFiles(dir);
var filenames
  = newBlockArr(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
prompt> ./parfilekmer -nl 4
```

Shared and Distributed-Memory Parallelism using forall, a distributed array, and command line options to indicate number of locales

PROCESSING FILES IN PARALLEL

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **See 'parfilekmer.chpl' in the repository**

- **Some things to try out with 'parfilekmer.html'**

```
# put more and bigger files into DataDir/ or set the config const dir to something else
chpl parfilekmer.html
./parfilekmer --dir="SomethingElse/"

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

POLL EV: LIST USAGE EXAMPLES

- length of list
- iterating over a list
- iterating over two lists at a time

```
list_1 = [1, 2, 3, 4]
list_2 = ['a', 'b', 'c']

for i, j in zip(list_1, list_2):
print(i, j)
```

- inserting and removing items from a list
- sum of the values in a list
- max value in a list

IMAGE PROCESSING EXAMPLE

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

- **See 'image_analysis_example/' subdirectory in the repository**
  - Coral reef diversity analysis written by Scott Bachman
  - Calls out to libpng to read and write PNG files
  - Uses distributed and shared memory parallelism

- **'image_analysis_example/README.md' explains how to compile and run it**

- **Some things to try out when running 'main'**

```
./main -nl 4 --inname=Roatan_benthic_r3_gray.png --outname=out1.png --radius=10

./main -nl 4 --inname=Roatan_benthic_r3_gray.png --outname=out2.png --radius=100

# Can also change the number of locales, but only up to the -N number given to salloc
```

What do you use programming for now?

Top

SORTING EXAMPLE

- **See 'diffusion.chpl' in the repository**
- **Some things to try out with 'diffusion.html'**

```
chpl kmer.html
./kmer

./kmer --k=10                                  # can change k
./kmer --infilename="kmer.chpl"                # can change the infilename
./kmer --k=10 --infilename="kmer.chpl"         # can change both
```

See https://github.com/mstrout/ChapelForPythonProgrammersFeb2023 for more info and for example code.

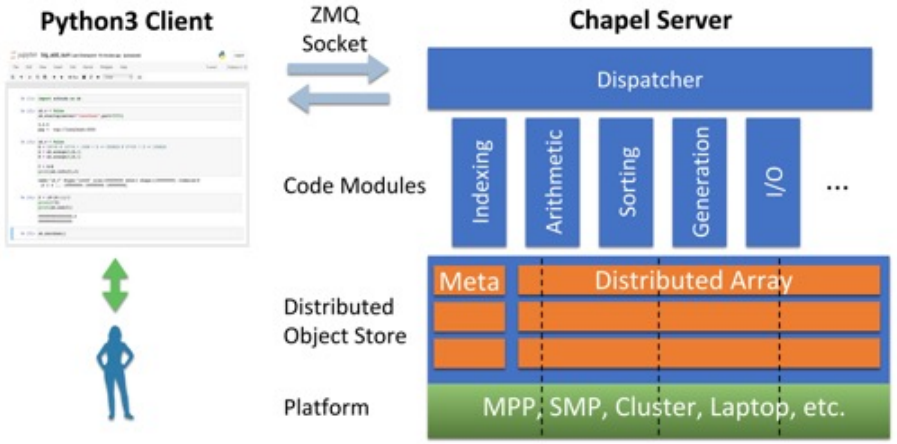
ARKOUDA IN ONE SLIDE

What is it?

- A Python library supporting a key subset of NumPy and Pandas for Data Science
- Implemented using a client-server model with Chapel as the server to support scalability
- Designed to compute results within the human thought loop (seconds to minutes on TB-scale arrays)
- ~35K lines of Chapel

Who did it?

- Mike Merrill, Bill Reus, et al., US DOD
- Open-source: https://github.com/Bears-R-Us/arkouda



Why Chapel?

- high-level language with C-comparable performance
- great distributed array support
- ports from laptop to supercomputer
- close to Pythonic—thus is readable for Python users who look under the hood

SOME SORT PERFORMANCE WITH ARKOUDA (FIXME)

What, if any, programs do you want to try with Chapel?

GPU SUPPORT IN CHAPEL

- **Generate code for GPUs**
  - Nascent support for NVIDIA
  - Exploring AMD and Intel support
- **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
- **Key concepts**
  - Using the 'locale' concept to indicate execution and data allocation on GPUs
  - 'forall' and 'foreach' loops will be converted to kernels
  - Arrays declared in 'on here.gpus[0]' blocks are allocated on the GPU
- **For more info...**
  - https://chapel-lang.org/docs/technotes/gpu.html

```
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());
```

OTHER CHAPEL EXAMPLES

- **Wavelet example by Jeremiah Corrado included in the github repository**
  - Slides and Code: https://github.com/mstrout/ChapelForPythonProgrammersFeb2023/tree/main/wavelet_example

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

- **Blog posts for Advent of Code**
  - https://chapel-lang.org/blog/index.html

- **Test directory in main repository**
  - https://github.com/chapel-lang/chapel/tree/main/test

TUTORIAL SUMMARY

- **Takeaways**
  - Chapel is a general-purpose programming language designed to leverage parallelism
  - It is being used in some large production codes
  - Our team is responsive to user questions and would enjoy having you participate in our community
- **How to get more help**
  - Ask us questions on discourse, gitter, or stack overflow
  - Also feel free to email me at michelle.strout@hpe.com
- **Engaging with the community**
  - Share your sample codes with us and your research community!
  - Join us at our free, virtual workshop in June, https://chapel-lang.org/CHIUW.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:

- Discourse: https://chapel.discourse.group/
- Gitter: https://gitter.im/chapel-lang/chapel
- Stack Overflow: https://stackoverflow.com/questions/tagged/chapel
- GitHub Issues: https://github.com/chapel-lang/chapel/issues