CHAPEL OVERVIEW, AND FUTURE OPPORTUNITIES AND CHALLENGES FOR CHAPEL

Michelle Mills Strout, Ben McDonald, Elliot Ronaghan, and Brad Chamberlain

SIAM PP22: Achieving Productivity at Scale with Chapel in User Applications
February 24, 2022
Chapel is truly a team effort—we're currently at 19 employees (+ a director), and we are hiring

Chapel Development Team at HPE

see: https://chapel-lang.org/contributors.html
Chapel is a parallel programming language that provides ease of 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.
OUTLINE

Scientific Computing Challenges
Data Analysis Example
How Applications are Using Chapel
Future Opportunities and Challenges
SCIENTIFIC COMPUTING CHALLENGES

• Steep learning curve to effectively achieve high performance
  • Distributed-memory parallelism across nodes (MPI)
  • Parallelism within a node (OpenMP, Pthreads, CUDA, ...)
  • Vectorization (intrinsics that are architecture specific)

• Preferred development model is on a laptop and then run on a cluster, cloud, or supercomputer

• Goal is to have ...
  • Ease of programming,
  • High performance, and
  • Portability

• Chapel achieves all three of these goals
use BlockDist;

config const m = 1000, alpha = 3.0;

const Dom = {1..m} dmapped ...;
var A, B, C: [Dom] real;

B = 2.0;
C = 1.0;
A = B + alpha * C;

forall (_, r) in zip(Updates, RAStream()) do T[r & indexMask].xor(r);

EASE OF PROGRAMMING AND HIGH PERFORMANCE
PORTABILITY

• On a laptop, cluster, or supercomputer (Shared-memory parallelism)

• On a cluster or supercomputer (Distributed-memory parallelism)
DATA ANALYSIS EXAMPLE
TRANSITIONING FROM LAPTOP TO SUPERCOMPUTER

- **Data Analysis Example**
  - Per file word count on all the files in a directory
  - Serial to threaded and distributed by using a *forall* over a parallel distributed array
  - Good scaling even for file I/O (below is for 100 files at 3MB each)
ANALYZING MULTIPLE FILES USING PARALLELISM

```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 { ... 
  while reader.readline(line) { 
    for word in line.split(" ") { 
      wordCount[word] += 1;
    }
  }
  ... }
```

prompt> chpl --fast word-count.chpl
prompt> ./word-count
prompt> ./word-count -nl 4

Shared and Distributed-Memory Parallelism using forall, a distributed array, and command line options to indicate number of locales
LAPTOP TO SUPERCOMPUTERS BASED ON ARRAY DISTRIBUTION

for loop: each iteration is executed serially by the current task
- predictable execution order, similar to conventional languages

forall loop: all iterations are executed by one or more tasks in no specific order
- implemented using one or more tasks, locally or distributed, as determined by the iterand expression

```
forall elem in myLocArr do ...  // task-level parallelism over local arrays
forall elem in myDistArr do ...  // distributed arrays use tasks on each locale owning part of the array
```

<table>
<thead>
<tr>
<th>Version of Parquet reader</th>
<th>1 Locale Performance</th>
<th>16 Locale Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.85 GiB/s</td>
<td>10.75 GiB/s</td>
</tr>
<tr>
<td>Parallel+Batch</td>
<td>7.46 GiB/s</td>
<td>23.26 GiB/s</td>
</tr>
</tbody>
</table>

benchmark uses 400 files of size 0.25 GiB each
HOW APPLICATIONS ARE USING CHAPEL

Refactoring existing codes into Chapel (~48K 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 (~16K lines of Chapel)

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

Chapel providing distributed parallelism for ML training (~8k lines of Chapel)

CrayAI: Distributed Machine Learning
Hewlett Packard Enterprise
FUTURE OPPORTUNITIES AND CHALLENGES FOR CHAPEL

• Generate code for GPUs (see Engin talk in MS79)
  • How will the compiler need to evolve? Will the language need to?

• Rearchitect the compiler
  • Shed cruft from research prototype days to harden the compiler
  • Reduce compile times
    – potentially via separate compilation / incremental recompilation?
  • Support interpreted / interactive Chapel programming

• Continue to optimize performance

• Release Chapel 2.0
  • guarantee backwards-compatibility for core language and library

• Foster a growing Chapel community

// HPCC Stream
// Variables stored on GPU
// forall’s are executed on GPU
on here.getChild(1) {
  var A, B, C: [1..n] real;
  const alpha = 2.0;

  forall b in B do b = 1.0;
  forall c in C do c = 2.0;

  forall a, b, c in zip(A, B, C) do
    a = b + alpha * c;
}
Chapel cleanly supports...

ease of programming,
high performance, and
portability

Chapel is being used for productive parallel applications at scale

- recent users have reaped its benefits in 10k–48k-line applications

Chapel provides clean ways to transition from laptop development to a cluster/supercomputer

The Chapel Development Team is

- ... at 19 people and is hiring!
- ... working on a number of exciting initiatives!
- ... looking forward to hearing from you!
CHIUW 2022 SUBMISSIONS DUE APRIL 15TH

The Chapel Parallel Programming Language

CHIUW 2022

The 9th Annual Chapel Implementers and Users Workshop

June 10, 2022
free and online in a virtual format

Call For Papers and Talks
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
THANK YOU

https://chapel-lang.org
@ChapelLanguage