CHAPEL PROGRAMMING LANGUAGE: OVERVIEW AND ROADMAP

Michelle Mills Strout
November 17, 2022
Chapel is truly a team effort

see: https://chapel-lang.org/contributors.html
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.
EASE OF PROGRAMMING AND HIGH PERFORMANCE

STREAM TRIAD: C + MPI + OPENMP

HPCC RA: MPI KERNEL

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);
...
PORTABILITY

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

```
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
```

• On a cluster or supercomputer (Distributed-memory parallelism)

```
prompt> chpl helloTaskPar.chpl
prompt> ./helloTaskPar --numLocales=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
...```
EXAMPLE CODE: 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.
**SCALING 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 10K files at 3MB each)
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

```plaintext
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 Node/Locale Performance</th>
<th>16 Node/Locale Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>for loop</td>
<td>0.85 GiB/s</td>
<td>10.75 GiB/s</td>
</tr>
<tr>
<td>forall loop</td>
<td>7.46 GiB/s</td>
<td>23.26 GiB/s</td>
</tr>
</tbody>
</table>

benchmark uses 400 Parquet files of size 0.25 GiB each
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
In Memoriam: Mike Merrill passed last week, and he will be greatly missed.

Mike was the originator and main developer of Arkouda.
CHAPEL ROADMAP

• Generate code for GPUs
  • Nascent support for NVIDIA
  • Exploring AMD and Intel support

• 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

// Stream
// Variables stored on GPU
// vector operations executed on GPU
config var n = 1_000_000, alpha = 0.01;

coforall loc in Locales on loc {
  coforall gpu in loc.gpus do on gpu {
    var A, B, C: [1..n] real;

    B = ...;
    C = ...;

    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–100k-line applications

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

The Chapel Development Team is
- ... working on a number of exciting initiatives!
- ... looking forward to hearing from you!
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