Chapel:
An Emerging Parallel Programming Language

Brad Chamberlain, Chapel Team, Cray Inc.
Emerging Technologies, SC13
November 20th, 2013
Q: Why doesn’t HPC have languages as enjoyable and productive as Python / Java / Matlab / (your favorite language here)?

A: We believe it’s due not to any particular technical challenge, but rather to a lack of sufficient...

...long-term efforts
...resources
...community will
...co-design between developers and users
...patience

Let’s change this!
What is Chapel?

- **An emerging parallel programming language**
  - Design and development led by Cray Inc.
    - in collaboration with academia, labs, industry
  - Initiated under the DARPA HPCS program

- **A work-in-progress**

- **Chapel’s overall goal: Improve programmer productivity**
  - Improve the *programmability* of parallel computers
  - Match or beat the *performance* of current programming models
  - Support better *portability* than current programming models
  - Improve the *robustness* of parallel codes
Chapel's Implementation

- Being developed as open source at SourceForge
- Licensed as BSD software

A Community Effort
- version 1.8 saw 19 developers from 8 organizations and 5 countries

Target Architectures:
- multicore desktops and laptops
- commodity clusters and the cloud
- HPC systems from Cray and other vendors
- in-progress: CPU+accelerator hybrids, manycore, …
**Multiresolution Design**: Support multiple tiers of features

- higher levels for programmability, productivity
- lower levels for greater degrees of control

Chapel language concepts

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine

- build the higher-level concepts in terms of the lower
- permit the user to intermix layers arbitrarily
Data Parallel Features

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine
STREAM Triad in Chapel

```chapel
const ProblemSpace = {1..m};

var A, B, C: [ProblemSpace] real;

A = B + alpha * C;
```

domains (first-class index sets)

promoted scalar operators

...and much more...
Base Language Features

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine
Tiled Row-Major Order Iterator

```plaintext
iter tiledRMO(D, tileSize) {
    const tile = {0..#tileSize, 0..#tileSize};

    for base in D by tileSize do
        for ij in D[tile + base] do
            yield ij;
}

for ij in tiledRMO({1..m, 1..n}, 2) do
    write(ij);
```

Prints:
(1,1)(1,2)(2,1)(2,2)(1,3)(1,4)(2,3)(2,4)...

CLU-style iterators

inferred types

algebra on domains
(index sets)
Task Parallel Features

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine
Coforall Loops

```
coforall t in 0..#numTasks {
    writeln(“Hello from task ”, t, “ of ”, numTasks);
} // implicit join of the numTasks tasks here
writeln(“All tasks done”);
```

Sample output:

```
Hello from task 2 of 4
Hello from task 0 of 4
Hello from task 3 of 4
Hello from task 1 of 4
All tasks done
```
Locality Control Features

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine
Chapel: Scoping and Locality

```plaintext
var i: int;
```
Chapel: Scoping and Locality

```chapel
var i: int;
on Locales[1] {
```
Chapel: Scoping and Locality

```chapel
var i: int;
on Locales[1] {
    var j: int;
}
```
Chapel: Scoping and Locality

```chapel
var i: int;
on Locales[1] {
  var j: int;
  coforall loc in Locales {
    on loc {
```

---

**Diagram:**

- **i**
- **j**
Chapel: Scoping and Locality

```chapel
var i: int;
on Locales[1] {
    var j: int;
coforall loc in Locales {
        on loc {
            var k: int;

            // within this scope, i,j,k can be referenced;
            // the implementation manages the communication
        }
    }
}
```
Domain Map Features

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine
Domain Maps

Domain maps are “recipes” that instruct the compiler how to map the global view of a computation...

\[ A = B + \alpha \cdot C; \]

...to the target locales’ memory and processors:
STREAM Triad: Chapel

const ProblemSpace = {1..m};

var A, B, C: [ProblemSpace] real;

A = B + alpha * C;
const ProblemSpace = {1..m};

var A, B, C: [ProblemSpace] real;

A = B + alpha * C;

No domain map specified => use default layout
- current locale owns all indices and values
- computation will execute using local processors only
STREAM Triad: Chapel (multilocale, blocked)

```
const ProblemSpace = {1..m}
    dmapped Block(boundingBox={1..m});

var A, B, C: [ProblemSpace] real;

A = B + alpha * C;
```
const ProblemSpace = {1..m}

dmapped Cyclic(startIdx=1);

var A, B, C: [ProblemSpace] real;

A = B + alpha * C;
Chapel’s Domain Map Philosophy

1. Chapel provides a library of standard domain maps
   ● to support common array implementations effortlessly

2. Advanced users can write their own domain maps in Chapel
   ● to cope with shortcomings in our standard library

3. Chapel’s standard domain maps are written using the same end-user framework
   ● to avoid a performance cliff between “built-in” and user-defined cases
Implementation Status -- Version 1.8.0 (Oct 2013)

Overall Status:
- Most features work at a functional level
  - some features need to be improved or re-implemented (e.g., OOP)
- Many performance optimizations remain
  - particularly for distributed memory (multi-locale) execution

This is a good time to:
- Try out the language and compiler
- Use Chapel for non-performance-critical projects
- Give us feedback to improve Chapel
- Use Chapel for parallel programming education
The Cray Chapel Team (Summer 2013)
Chapel…

…is a collaborative effort — join us!
Chapel: the next five years

● Harden Prototype to Production-grade
  ● Performance optimizations
  ● Add/Improve features that are lacking

● Target more complex/modern compute node types
  ● e.g., CPU+GPU, Intel MIC, …

● Continue to grow the user and developer communities
  ● Including nontraditional circles: desktop parallelism, “big data”
  ● Transition Chapel from Cray-controlled to community-governed

● Grow the team at Cray
  ● we’ve just hired four new developers
  ● we’re currently hiring for a manager position
Chapel at SC13

- **Emerging Technologies Booth** *(tomorrow)*
  - Booth #3547: staffed by Chapel team members; poster and handouts

- **Poster** *(Tues @ 5:15)*: *Towards Co-Evolution of Auto-Tuning and Parallel Languages*
  - Posters Session: Ray Chen (University of Maryland)

- **Talk** *(Tues @ 3:20)*: *Hierarchical Locales: Exposing the Node Architecture in Chapel*
  - KISTI booth (#3713): Sung-Eun Choi (Cray Inc.)

- **Chapel Lightning Talks BoF** *(Wed @ 12:15)*
  - 5-minute talks on education, MPI-3, Big Data, Autotuning, Futures, MiniMD

- **Talk** *(Wed @ 4:30)*: *Chapel, an Emerging Parallel Language*
  - HPC Impact Theatre (booth #3947): Brad Chamberlain (Cray Inc.)

- **Happy Hour** *(Wed @ 5pm)*: *4th annual Chapel Users Group (CHUG) Happy Hour*
  - Pi Bar (just across the street at 1400 Welton St): open to public, dutch treat

- **HPC Education** *(Thus @ 1:30pm)*: *High-Level Parallel Programming Using Chapel*
  - David Bunde (Knox College) and Kyle Burke (Colby College)
For More Information: Online Resources

Chapel project page: http://chapel.cray.com
● overview, papers, presentations, language spec, …

Chapel SourceForge page: https://sourceforge.net/projects/chapel/
● release downloads, public mailing lists, code repository, …

Mailing Aliases:
● chapel_info@cray.com: contact the team at Cray
● chapel-announce@lists.sourceforge.net: announcement list
● chapel-users@lists.sourceforge.net: user-oriented discussion list
● chapel-developers@lists.sourceforge.net: developer discussion
● chapel-education@lists.sourceforge.net: educator discussion
● chapel-bugs@lists.sourceforge.net: public bug forum
Chapel at SC13

- **Emerging Technologies Booth** *(tomorrow)*
  - Booth #3547: staffed by Chapel team members; poster and handouts
- **Poster** *(Tues @ 5:15)*: *Towards Co-Evolution of Auto-Tuning and Parallel Languages*
  - Posters Session: Ray Chen (University of Maryland)
- **Talk** *(Tues @ 3:20)*: *Hierarchical Locales: Exposing the Node Architecture in Chapel*
  - KISTI booth (#3713): Sung-Eun Choi (Cray Inc.)
- **Chapel Lightning Talks BoF** *(Wed @ 12:15)*
  - 5-minute talks on education, MPI-3, Big Data, Autotuning, Futures, MiniMD
- ✓ **Talk** *(Wed @ 4:30)*: *Chapel, an Emerging Parallel Language*
  - HPC Impact Theatre (booth #3947): Brad Chamberlain (Cray Inc.)
- **Happy Hour** *(Wed @ 5pm)*: 4th annual Chapel Users Group (CHUG) Happy Hour
  - Pi Bar (just across the street at 1400 Welton St): open to public, dutch treat
- **HPC Education** *(Thus @ 1:30pm)*: *High-Level Parallel Programming Using Chapel*
  - David Bunde (Knox College) and Kyle Burke (Colby College)