What is Chapel?

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

- **Overall goal:** Improve programmer productivity

- A work-in-progress
Chapel's Implementation

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

**Target Architectures:**
- Cray architectures
- multicore desktops and laptops
- commodity clusters
- systems from other vendors
- (in-progress: CPU+accelerator hybrids, manycore, …)
Outline

✓ Chapel Context

➢ Chapel Background for today’s talks
● Project Information
begin:

begin foo();  // create a task to run foo
bar();  // original task continues on

single:

var area1$, area2$: single real;
begin area1$ = computeArea(shape1);
begin area2$ = computeArea(shape2);
doSomethingElse();

const totalArea = area1$ + area2$
Data Parallelism

\[
\text{const } D = \{1..n\} \text{ dmapped Cyclic(startIdx=1);} \\
\text{var A, B, C: [D] real;} \\
\text{forall (a,b,c) in zip (A,B,C) do} \\
\text{a = b + alpha * c;}
\]

High-level features implemented…
• in Chapel
• using lower-level features
• by end-users

\[
\text{var area1$, area2$: single real;} \\
\text{begin area1$ = computeArea(shape1);} \\
\text{begin area2$ = computeArea(shape2);} \\
\text{doSomethingElse();}
\]

\[
\text{const totalArea} = \text{area1$} + \text{area2$}
\]
Chapel and Education

- When teaching parallel programming, it’s important to cover:
  - data parallelism
  - task parallelism
  - concurrency
  - synchronization
  - locality/affinity
  - deadlock, livelock, and other pitfalls
  - performance tuning
  - ...

- But there hasn’t been a good language out there...
  - for teaching *all* of these things
  - for teaching some of these things well at all
  - *until now:* We believe Chapel can potentially play a crucial role here

(see [http://chapel.cray.com/education.html](http://chapel.cray.com/education.html) for more information)
Compiling Chapel

Chapel Source Code → chpl → Chapel Executable

Standard Modules (in Chapel)
Chapel Compiler Architecture

Chapel Source Code → Chapel-to-C Compiler → Generated C Code → Standard C Compiler & Linker → Chapel Executable

- Chapel Compiler
- Chapel-to-C Compiler
- Generated C Code
- Standard C Compiler & Linker
- Chapel Executable

- Standard Modules (in Chapel)
- Internal Modules (in Chapel)
- Runtime Support Library (in C)
  - Tasks/Threads
  - Communication
  - Memory
  - ...
**Runtime libraries** implement low-level features
- e.g., tasking, communication, memory, …
- we refer to these feature areas as *layers*
- the lowest level of Chapel software stack

Chapel Compiler Architecture
Runtime Communication Layer

Chapel Runtime Support Library (in C)

Communication

- none (single locale)
- gasnet
- ugni
- mpi
  - GASNet (universal)
  - Cray uGNI (Cray networks)
  - MPI-3 (universal)
Runtime Communication Layer

Chapel Runtime Support Library (in C)

Communication

- none (single locale)
- gasnet (universal)
- ugni (Cray networks)
- mpi (universal)
- MPI-3 (universal)

An exciting work in progress
- One of today’s talks
Outline

✓ Chapel Context
✓ Chapel Background for today’s talks
➢ Project Information
Chapel...

...is a collaborative effort — join us!
“I Like Chapel, how can I help?”

- **Let people know that you like it and why**
  - your colleagues
  - your employer/institution
  - Cray leadership (e.g., mention it at the Cray booth this week)

- **Help us evolve it from prototype to production**
  - contribute back to the source base
  - collaborate with us
  - help fund the effort
  - help us transition from “How will Cray make Chapel succeed?” to “How can we as a community make Chapel succeed?”
Resources For After Today

**Chapel project page:** [http://chapel.cray.com](http://chapel.cray.com)
- papers, presentations, tutorials, language spec, ...

**Chapel SourceForge page:** [https://sourceforge.net/projects/chapel/](https://sourceforge.net/projects/chapel/)
- release downloads, code repository, public mailing lists, ...

**IEEE TCSC Blog Series:**
- *Myths About Scalable Parallel Programming Languages*

**Mailing Lists:**
- chapel_info@cray.com:
- chapel-users@lists.sourceforge.net: user-oriented discussion list
- chapel-developers@lists.sourceforge.net: dev.-oriented discussion
- chapel-education@lists.sourceforge.net: educator-oriented discussion
- chapel-bugs@lists.sourceforge.net chapel_bugs@cray.com : public/private bug forum
Chapel at SC13

● **Emerging Technologies Booth** (all week)
  ● 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)