# Chapel Overview

Brad Chamberlain, Chapel Team, Cray Inc.

SC11: November 16th, 2011

# What is Chapel?

- A new parallel programming language
  - Design and development led by Cray Inc.
    - In collaboration with academics, labs, industry
  - Initiated under the DARPA HPCS program
- **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
- A work-in-progress

# Chapel's Implementation

- Being developed as open source at SourceForge
- Licensed as BSD software
- **Target Architectures:**
  - multicore desktops and laptops
  - commodity clusters
  - Cray architectures
  - systems from other vendors
  - (in-progress: CPU+accelerator hybrids, manycore, …)

# Chapel's Origins

- **HPCS**: High Productivity Computing Systems
  - Overall goal: Raise high-end user productivity by 10x

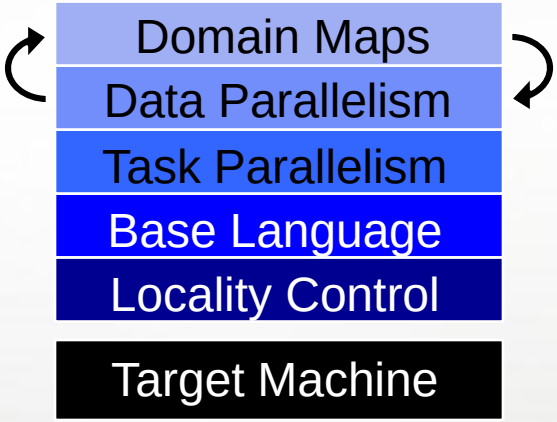
    ***Productivity = Performance + Programmability + Portability + Robustness***
- **Phase II:** Cray, IBM, Sun (July 2003 – June 2006)
  - Goal: Propose new productive system architectures
  - Each vendor created a new programming language
    - **Cray:** Chapel
    - **IBM:** X10
    - **Sun:** Fortress
- **Phase III:** Cray, IBM (July 2006 – )
  - Goal: Develop the systems proposed in phase II
  - Each vendor implemented a compiler for their language
    - Sun also continued their Fortress effort without HPCS funding

# Chapel's Multiresolution Design

***Multiresolution Design:*** Support multiple tiers of features

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

*Chapel language concepts*




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

# Sample Chapel Features

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

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

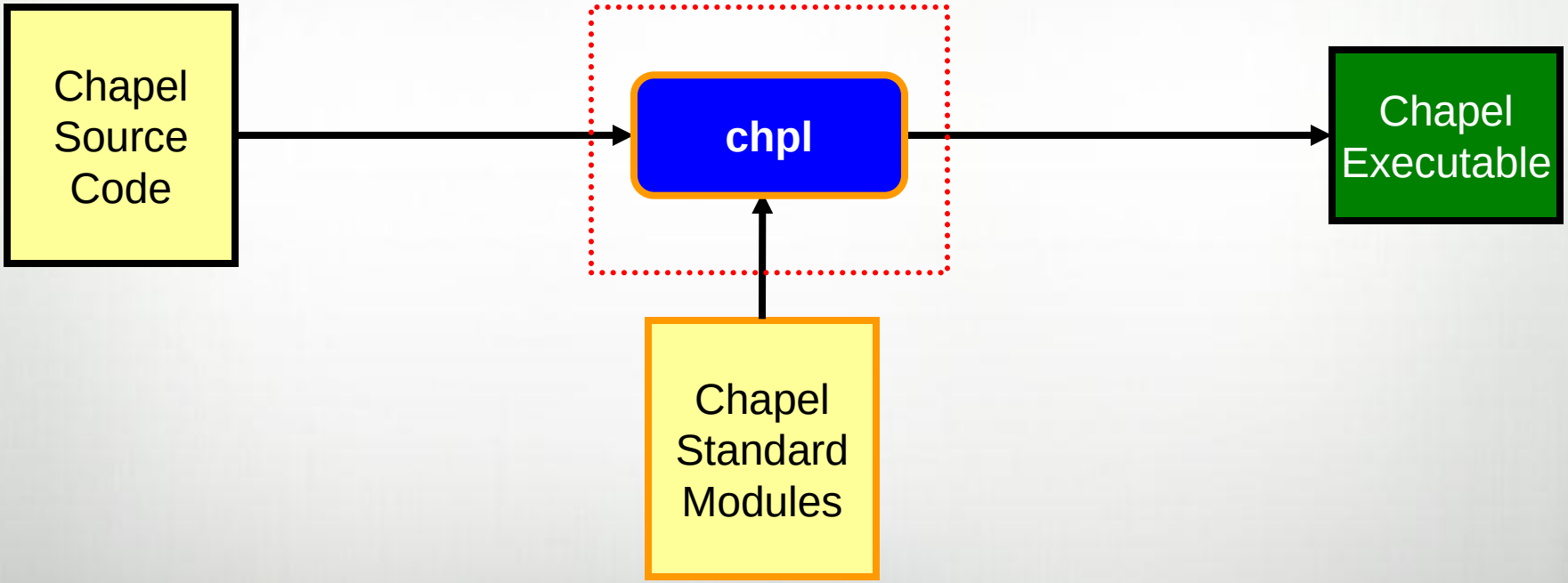
```
var buffer$: [0..numElts] sync real;
cobegin {
  on Locales[1] do producer(buffer$);
  on A[i]       do consumer(buffer$);
}
```

*Chapel language concepts*

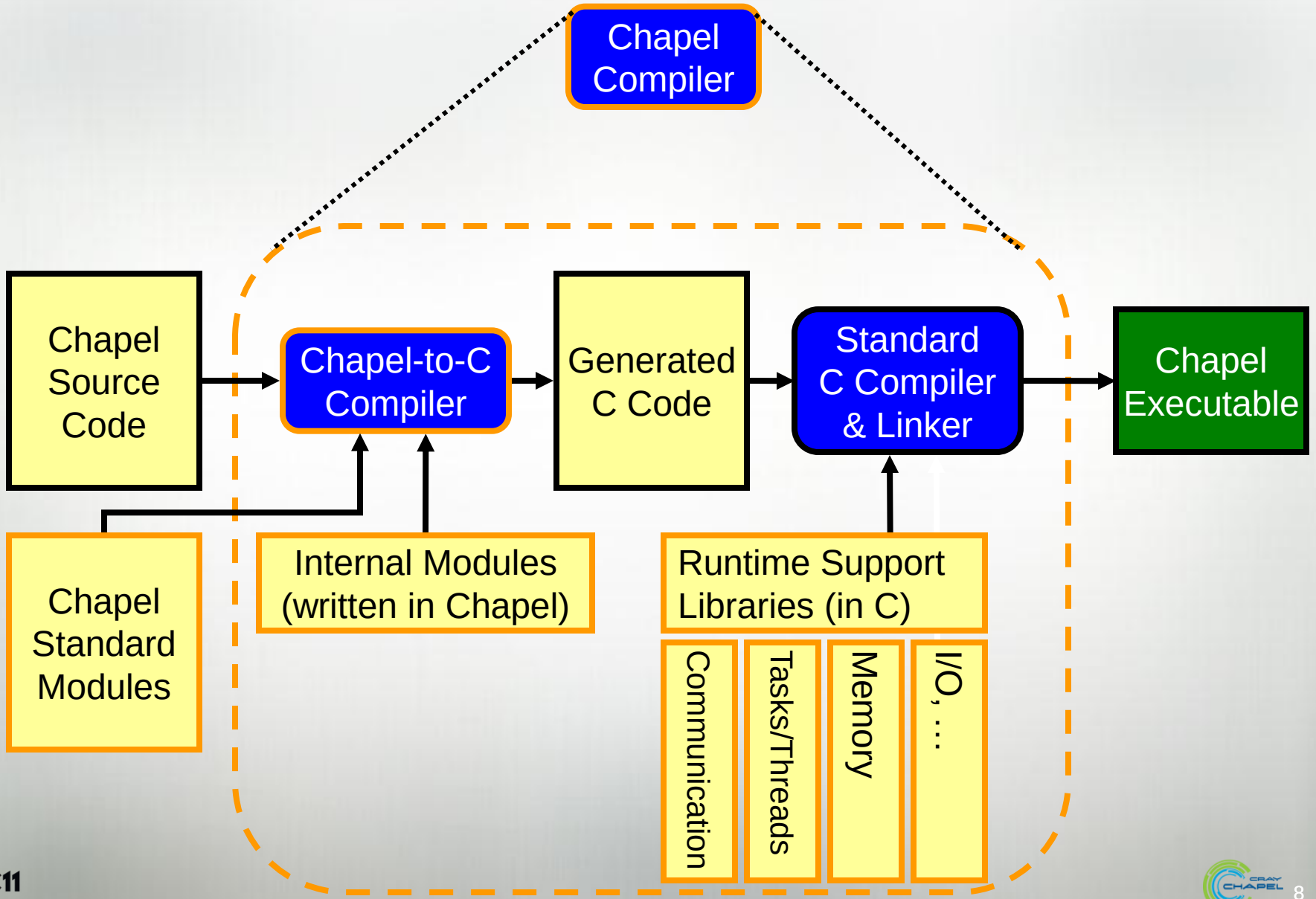
- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control

Target Machine

# Compiling Chapel



# Chapel Compiler Architecture



# Implementation Status -- Version 1.4.0

**In a nutshell:**

- Most features work at a functional level
- Many performance optimizations remain

**This is a good time to:**

- Try out the language and compiler
- Give us feedback to improve Chapel
- Use Chapel for non-performance-critical projects
- Use Chapel for parallel programming education

**In evaluating the language:**

- Try to judge it by how it should *ultimately* perform rather than how it does today
  - lots of low-hanging fruit remains, as well as some challenges

# Chapel at SC11 (see chapel.cray.com/events.html for details)

- **Mon:** full-day tutorial
- **Mon:** 2nd annual CHUG happy hour/meet-up
- **Tues:** HPC Challenge BoF (12:15-1:15)
- ➢ **Wed:** Chapel Lightning Talks BoF (12:15-1:15)
- **Thurs:** "Punctuated Equilibrium at Exascale" Panel (5:30-7:00)
- **Fri:** half-day tutorial

- **T-Th:** Chapel posters in PGAS booth, Chapel team members staffing (T 2-4, W 10-12, W 4-6, Th 10-12)

# Chapel Team's Next Steps

- Continue to improve performance and add missing features
- Expand the set of codes that we are studying
- Expand the set of architectures that we can target effectively
- Support the release
- Continue to support collaborations and seek out new ones

- Determine Chapel's future after HPCS ends (October 2012)

# Chapel 5-year Plan: Key Components

- **Advisory Board**
  - help steer Chapel team's priorities on a regular basis
    - performance vs. features vs. a mix of both
    - which optimizations and features to prioritize
    - which benchmarks, idioms to focus on
- **Agile milestones rather than *a priori***
  - dynamically react to community's needs, R&D challenges
- **Improve openness of project, transition to community**
- **Unified Chapel reporting**
  - rather than reporting to several programs, Chapel is the program
  - reduces reporting burden, permitting team to focus more on work
  - brings those interested in Chapel to a single meeting

# If you end up thinking "I Like Chapel, how can I help?"

- **Let people know that you like it and why**
  - your colleagues
  - your employer/institution
  - Cray leadership (stop by the Cray booth this week)

- **Help us evolve it from prototype to production**
  - contribute back to the source base
  - collaborate with us
  - help fund us to grow the team
  - help us get from "How will Cray make Chapel succeed?" to "How can we as a community make Chapel succeed?"

# Join Our Growing Community

- Cray:


Brad Chamberlain


Sung-Eun Choi


Greg Titus


Vass Litvinov


Tom Hildebrandt

- External Collaborators:


Albert Sidelnik
(UIUC)


Jonathan Turner
(CU Boulder)


Kyle Wheeler
(Sandia)

•••

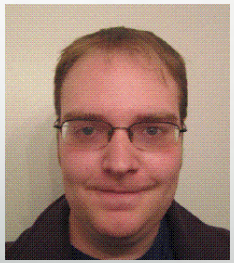


You? Your Friend/Student/ Colleague?

- Interns:


Jonathan Claridge
(UW)


Hannah Hemmaplardh
(UW)


Andy Stone
(Colorado State)


Jim Dinan
(OSU)


Rob Bocchino
(UIUC)


Mackale Joyner
(Rice)

•••

# Featured Collaborations (see chapel.cray.com/collaborations.html for details)

- **Tasking using Qthreads:** Sandia (Rich Murphy, Kyle Wheeler, Dylan Stark)
  - **paper at CUG, May 2011**
- **Interoperability using Babel/BRAID:** LLNL (Tom Epperly, Adrian Prantl, et al.)
  - **paper at PGAS, Oct 2011**
- **Dynamic Iterators:**
- **Bulk-Copy Opt:**
- **Parallel File I/O:**

  U Malaga (Rafael Asenjo, Maria Angeles Navarro, et al.)
  - **paper at ParCo, Aug 2011**
- **Improved I/O & Data Channels:** LTS (Michael Ferguson)
- **CPU-GPU Computing:** UIUC (David Padua, Albert Sidelnik, Maria Garzarán)
  - **tech report, April 2011**
- **Interfaces/Generics/OOP:** CU Boulder (Jeremy Siek, Jonathan Turner)
- **Tasking over Nanos++:** BSC/UPC (Alex Duran)
- **Tuning/Portability/Enhancements:** ORNL (Matt Baker, Jeff Kuehn, Steve Poole)
- **Chapel-MPI Compatibility:** Argonne (Rusty Lusk, Pavan Balaji, Jim Dinan, et al.)

# Collaboration Ideas (see chapel.cray.com/collaborations.html for details)

- memory management policies/mechanisms
- dynamic load balancing: task throttling and stealing
- parallel I/O and checkpointing
- exceptions; resiliency
- application studies and performance optimizations
- index/subdomain semantics and optimizations
- targeting different back-ends (LLVM, MS CLR, …)
- runtime compilation
- library support
- tools: debuggers, performance analysis, IDEs, interpreters, visualizers
- database-style programming
- autotuning
- (your ideas here…)

# For More Information

**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 Lists:**

- chapel_info@cray.com: contact the team
- 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: public bug forum
- chapel_bugs@cray.com: private bug mailing list