CSE 590o: Chapel

Brad Chamberlain

Steve Deitz

Chapel Team

University of Washington

September 26, 2007

Outline

- Context for Chapel
- This Seminar
- Chapel Compiler

Chapel

Chapel: a new parallel language being developed by Cray Inc.

Themes:

- general parallelism
  - data-, task-, nested parallelism using *global-view* abstractions
  - general parallel architectures
- locality control
  - data distribution
  - task placement (typically data-driven)
- reduce gap between mainstream and parallel languages
  - object-oriented programming (OOP)
  - type inference and generic programming

Chapel's Setting: HPCS

HPCS: High *Productivity* Computing Systems (DARPA *et al.*)

- Goal: Raise HEC user productivity by 10× for the year 2010
- Productivity = Performance
  + Programmability
  + Portability
  + Robustness

- **Phase II**: Cray, IBM, Sun (July 2003 – June 2006)
  - Evaluated the entire system architecture's impact on productivity…
    - processors, memory, network, I/O, OS, runtime, compilers, tools, …
    - …and new languages:

      Cray: Chapel IBM: X10 Sun: Fortress

- **Phase III**: Cray, IBM (July 2006 – 2010)
  - Implement the systems and technologies resulting from phase II
  - (Sun also continues work on Fortress, without HPCS funding)

Chapel and Productivity

Chapel's Productivity Goals:

- vastly improve programmability over current languages/models
  - writing parallel codes
  - reading, modifying, porting, tuning, maintaining them
- support performance at least as good as MPI
  - competitive with MPI on generic clusters
  - better than MPI on more capable architectures
- improve portability compared to current languages/models
  - as ubiquitous as MPI, but with fewer architectural assumptions
  - more portable than OpenMP, UPC, CAF, …
- improve code robustness via improved semantics and concepts
  - eliminate common error cases altogether
  - better abstractions to help avoid other errors

Chapel Design Philosophies

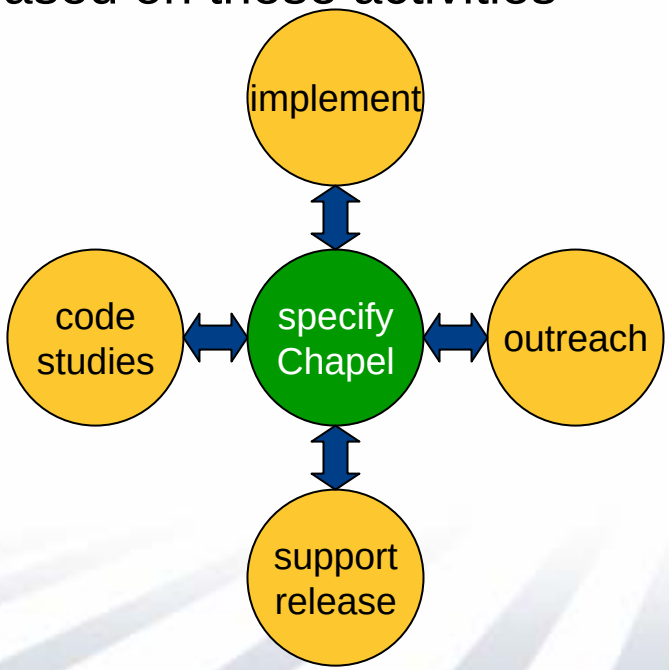
- A research project…
  - …but intentionally broader than an academic project would tend to be
    - due to the belief that generality requires a broad feature set
    - to create a space for broad community participation/collaboration
- Nurture within Cray, then turn over to community
  - currently releasing to small set of “friendly” users
  - hope to do public release in late 2008
- Borrow when it makes sense, innovate elsewhere
  - interplay between borrowed concepts is where many challenges lie
- Language design as art / beauty in eye of beholder
  - many of our decisions have been subjective
  - some of them, even we don’t like

Chapel Influences

- **ZPL, HPF:** data parallelism, index sets, distributed arrays, aggregate operations (see also APL, NESL, Fortran90)
- **Cray MTA C/Fortran:** task parallelism, lightweight synch.
- **CLU:** iterators/generators (see also Ruby, Python, C#?)
- **ML, Scala, Matlab, Perl, Python:** latent types
- **Java, C#:** OOP, type safety
- **C++:** generic programming/templates
- **C, Modula, Ada:** syntax

Chapel Work

- Chapel Team's Focus:
  - specify Chapel syntax and semantics
  - implement prototype Chapel compiler
  - code studies of benchmarks, applications, and libraries in Chapel
  - community outreach to inform and learn from users, colleagues
  - support users evaluating our preliminary releases
  - refine language based on these activities



This Seminar

- **Goals:**
  - Introduce the UW community to Chapel
  - Solicit feedback about Chapel from…
    - …programming language / compiler / parallel programming groups
    - …potential users
  - Identify opportunities for collaboration

- **Structure:**
  - **week 1:** context
  - **week 2:** whole-language overview
  - **weeks 3-9:** deep dives into feature sets
    - definition, rationale
    - open questions, opportunities for feedback
  - **week 10:** grab-bag/open-ended

Seminar Outline

Date	Topic	Reading	Facilitator
Sep 26	Chapel context, Seminar goals	--	Chamberlain
Oct 3	Chapel overview	IJHPCA paper	Chamberlain
Oct 10	Language fundamentals	Ch 6-13, 17	Chamberlain
Oct 17	OOP & generics	Ch 14-16, 21	Deitz
Oct 24	Ranges, domains, arrays	Ch 18-19, 24	Chamberlain
Oct 31	TBD (may be cancelled)		
Nov 7	Iterators	Ch 20	Deitz
Nov 14	Task parallelism, synchronization	Ch 22	Deitz
Nov 21	NO MEETING (Thanksgiving)	--	--
Nov 28	Distributions, locality	Ch 23	Chamberlain
Dec 5	Open issues / grab-bag	--	Chamberlain

Ground Rules

- For us:
  - be open, honest about project status – avoid sales pitches
    - what is “solved”
    - where we believe we have a solution
    - where we have a promising path ahead of us
    - where large open questions remain
  - take criticism constructively

- For you:
  - tell us your thoughts, reactions, insights, and criticism
  - realize that some things would be difficult to change at this point
  - if session times out, please follow up over email

Who we are

- Our current team (sorted by time on project):
  - Brad Chamberlain (bradc@cray.com)
  - Steve Deitz (deitz@cray.com)
  - Mary Beth Hribar
  - David Iten
  - Samuel Figueroa

- Current academic collaborations:
  - **Vikram Adve & Robert Bocchino (UIUC):** software transactional memory for distributed memory computers
  - **Franz Franchetti (CMU):** SPIRAL back-end targeting Chapel to leverage its portability
  - **<Your Name Here?> (UW):** …

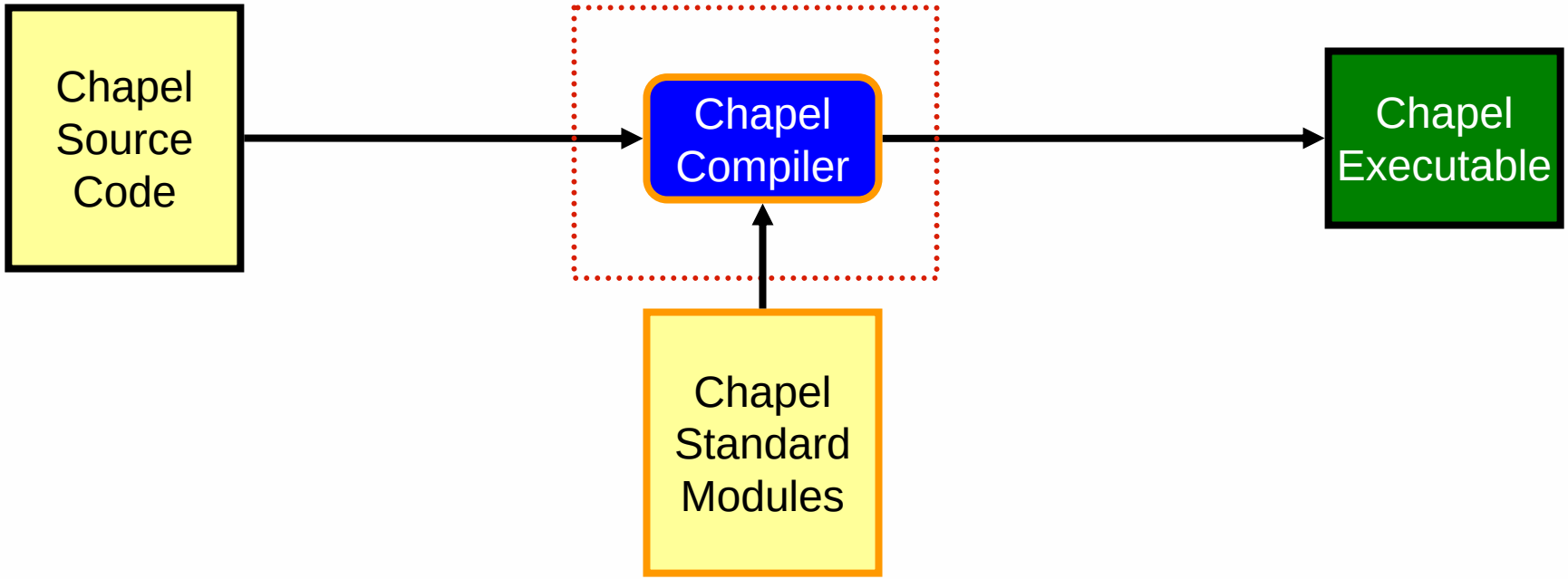
Who are you?

- Name
- Department / Advisor
- General Research Interests
- Specific Interests in Chapel / this seminar

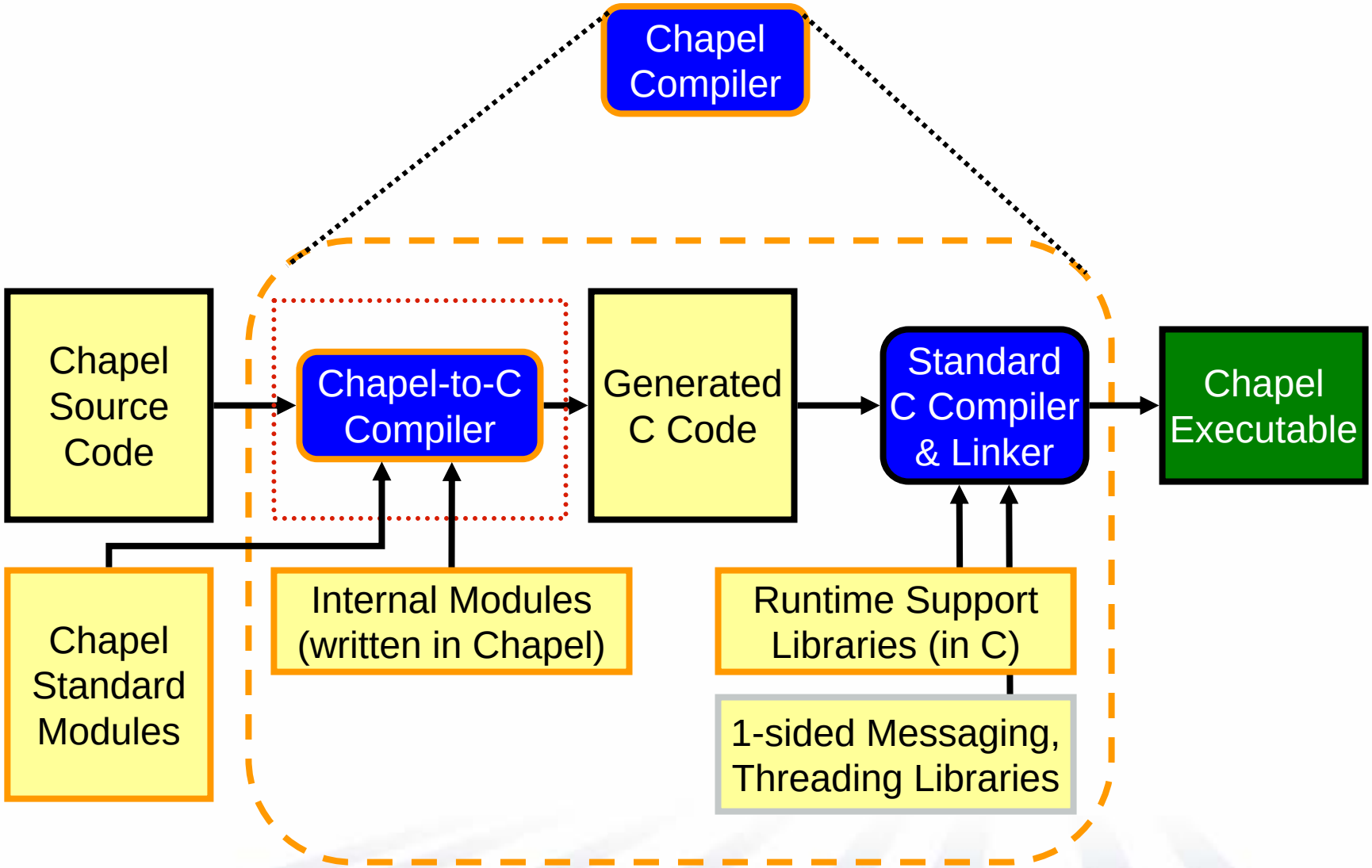
“How do I earn credit for this course?”

- Participation in discussions a must
  - we should have some sense of who you are by end of quarter
- Remainder open to negotiation; choose one of:
  - program some parallel algorithm of interest in Chapel
    - submit code plus short report
    - track bugs, workarounds, feature requests
  - facilitate next week’s session
  - co-facilitate a language topic session
    - present survey of a week’s concept in other languages
    - help lead discussion on a Chapel topic
  - submit written comments/suggestions on the language specification
  - propose your own idea
- Taking for two credits? Do 2 of these, or 1 in more depth
- Mail brief proposal of how you would like to earn credit for the seminar to bradc@cray.com by next week’s session

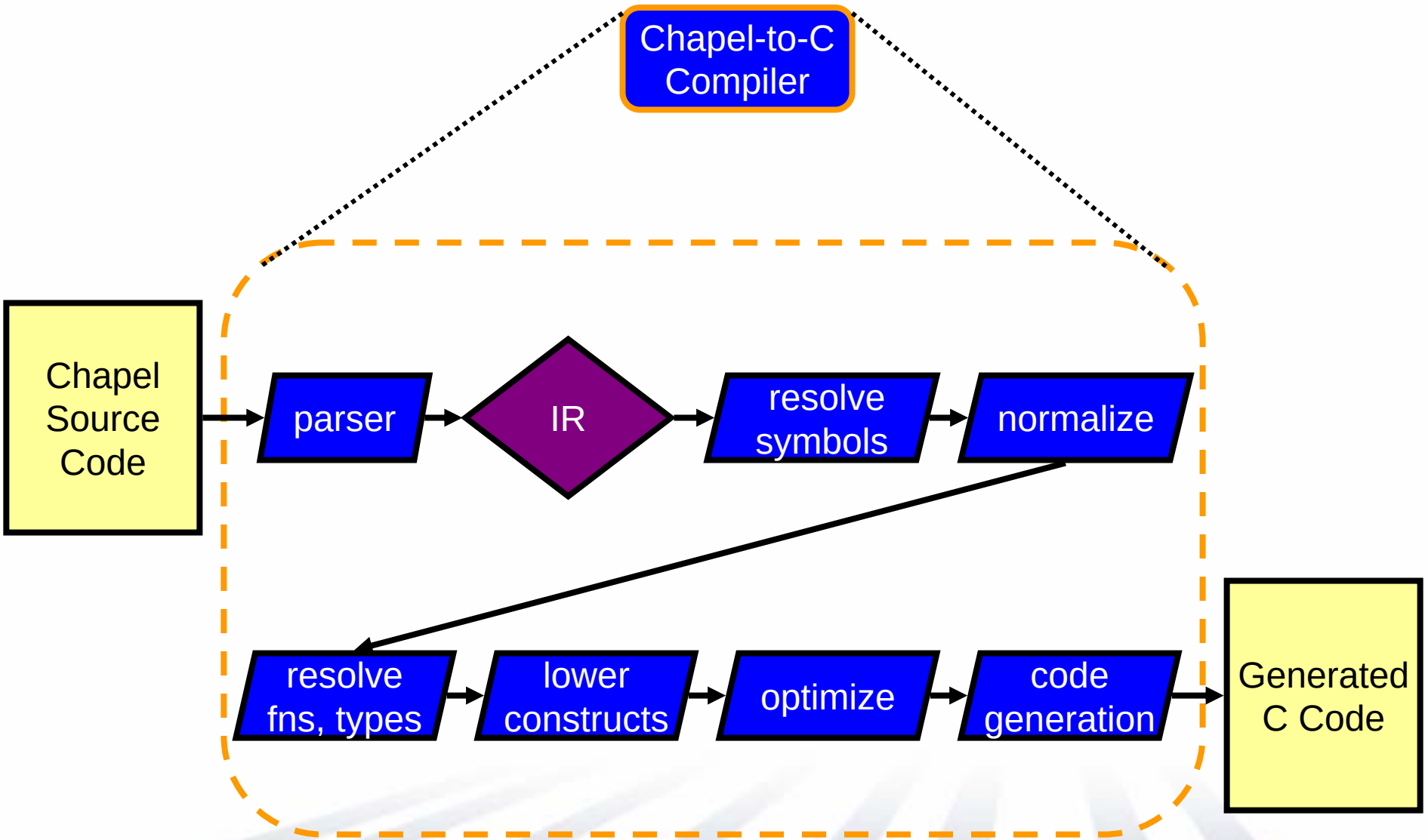
Compiling Chapel



Chapel Compiler Architecture



Chapel-to-C Architecture



Prototype Compiler Status

- **Features:** enough there to experiment with
  - **Base language features:** in decent shape
  - **Task parallel features:** implemented naively using pthreads for one *locale* (multicore processor, SMP node, etc.)
  - **Data parallel features:** implemented, but do not currently generate parallelism
  - **Multi-locale (dist. memory) features:** essentially unimplemented
- **Performance:** has not been a primary concern to date
  - **execution speed:** tuned for some 1D idioms, not much for others
  - **memory:** avoids large temporary variables, but leaks smaller stuff
- **Getting Access:** need to fill out the user agreement
  - will make an installation available on CSE machines
  - will make a downloadable copy available to others
- **Help us Improve:** if you use the prototype compiler, track…
  - **bugs:** chapel_bugs@cray.com
  - **questions, feature requests:** chapel_info@cray.com

TODOs for next week

- Yours:
  - read IJHPCA paper (link to paper on course web)
  - mail proposal for earning credit to bradc@cray.com

- Ours:
  - set up mailing list
  - update course web with schedule, readings
  - install Chapel prototype compiler

For More Information…

http://www.cs.washington.edu/education/courses/590o/07au

http://chapel.cs.washington.edu

chapel_info@cray.com

bradc@cray.com

deitz@cray.com

Questions?