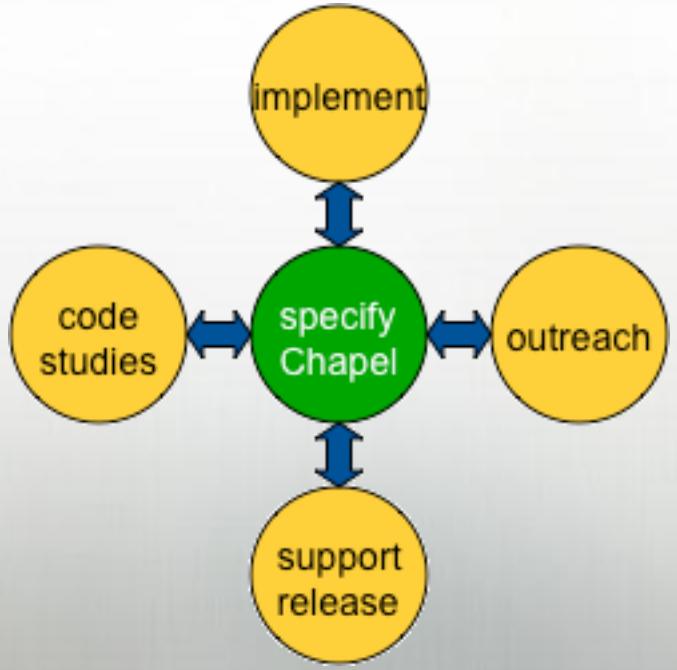
CRAY
THE SUPERCOMPUTER COMPANY

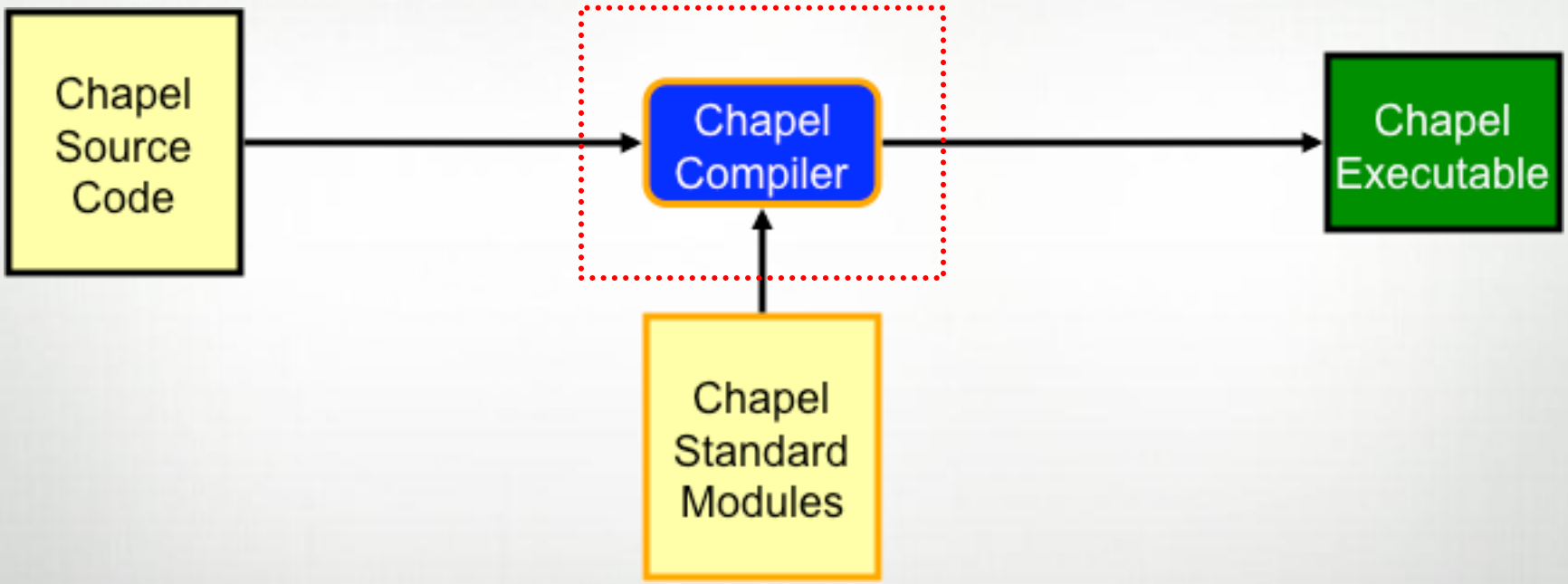
Chapel: Project Summary

Chapel Work

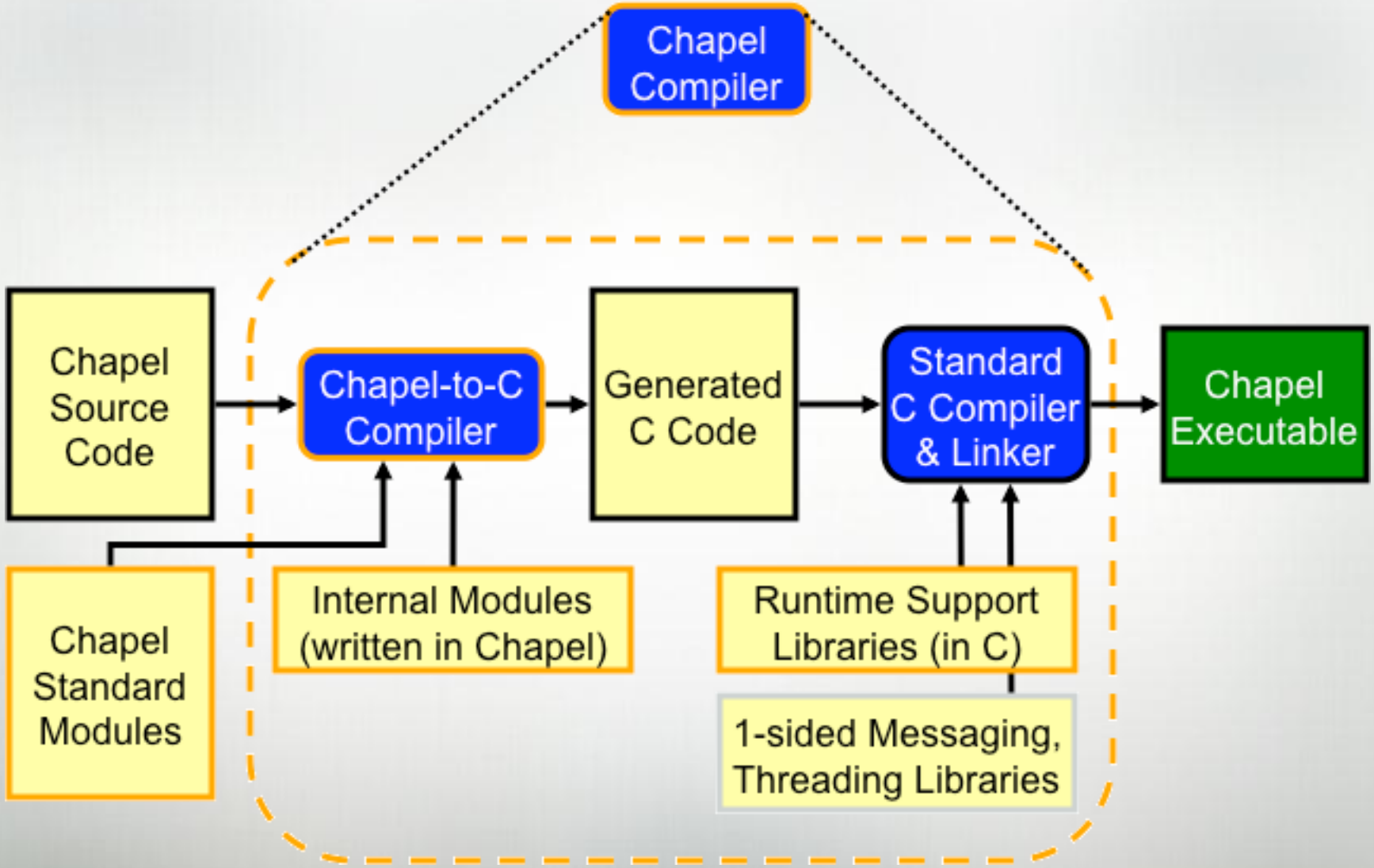
- Chapel Team's Focus:
  - specify Chapel syntax and semantics
  - implement open-source prototype compiler for Chapel
  - perform code studies of benchmarks, apps, and libraries in Chapel
  - do community outreach to inform and learn from users/researchers
  - support users of code releases
  - refine language based on all these activities



Compiling Chapel



Chapel Compiler Architecture



Chapel Version 1.2

- Features
  - Open source at http://sourceforge.net/projects/chapel/
  - Distributed under the BSD Open Source license
  - Supports Linux/Unix, Mac, Cygwin, Cray platforms
- Contents
  - Compiler, runtime, standard modules, third-party libraries
  - Language spec, quick reference, numerous examples

Implementation Status

- In a nutshell…
  - Most features working at a functional level
  - Many performance optimizations remain
- This is a good time to…
  - Try the language
  - Give us feedback to make the language better
  - Use Chapel for parallel programming courses
  - Use Chapel for non-performance critical coding
- In evaluating the language…
  - Evaluate it based on how it should be able to perform rather than how it does today

Current Collaborations

- **Notre Dame/ORNL** (Peter Kogge, Srinivas Sridharan, Jeff Vetter) Asynchronous software transactional memory over distributed memory
- **UIUC** (David Padua, Albert Sidelnik, Maria Garzaran) CPU-GPU computing
- **BSC/UPC** (Alex Duran) Chapel over Nanos++ user-level tasking
- **U. Malaga** (Rafa Asenio, Maria Gonzales, Rafael Larossa) Parallel file I/O
- **U. Colorado** (Jeremy Siek, Jonathan Turner) Interfaces and generics
- **PNNL/CASS-MT** (John Feo, Daniel Chavarria) Hybrid computing in Chapel; Cray XMT performance tuning; ARMCI port
- **ORNL** (David Bernholdt *et al.*, Steve Poole *et al.*) Code studies – Fock matrices, MADNESS, Sweep3D, coupled models, ...
- **Berkeley** (Dan Bonachea, Paul Hargrove *et al.*) Efficient GASNet support; collective communication
- **U. Oregon/Paratools Inc.** (Sameer Shende) Performance analysis with Tau
- (your name here?)

Collaboration Opportunities (see chapel.cray.com for more details)

- memory management policies/mechanisms
- dynamic load balancing: task throttling and stealing
- parallel I/O and checkpointing
- exceptions; resiliency
- language interoperability
- 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
- (your ideas here…)

Chapel and Education

- If I were teaching a parallel programming class, I'd want to teach about:
  - data parallelism
  - task parallelism
  - concurrency
  - synchronization
  - locality/affinity
  - deadlock, livelock, and other pitfalls
  - performance tuning
  - …
- I don't think there's a good language out there…

  …for teaching *all* of these things

  …for teaching some of these things at all

  …until now: I think Chapel has the potential to play a crucial role here

Our Next Steps

- Expand our set of supported distributions
- Continue to improve performance
- Continue to add missing features
- Expand the set of codes that we are studying
- Expand the set of architectures that we are targeting
- Support the public release
- Continue to support collaborations and seek out new ones
- Continue to expand our team

Summary

Chapel strives to greatly improve Parallel Productivity

via its support for…

- …general parallel programming
- …global-view abstractions
- …control over locality
- …multiresolution features
- …modern language concepts and themes

http://chapel.cray.com/ ◆ http://sourceforge.net/projects/chapel ◆ chapel_info@cray.com