Chapel: Compiler Overview

Steve Deitz
Cray Inc.
Outline

- Chapel Compiler System Overview
- Version 0.9 Release and Status
Prototype Compiler Development Strategy

- Start development within Cray under HPCS
- Initial releases to select users
- First public release November 2008
- Second public release April 2009
  - Migrated to SourceForge
  - Major step in opening development
- Turn over to community when ready
Chapel-to-C compiler for portability
  - C++ compiler generates strict C code
  - Tested against GCC and several vendor’s compilers

Link against threading and communication libraries
  - Default threading layer on most platforms: pThreads
  - Default communication layer on most platforms: GASNet

Use many standard and internal Chapel modules
Compiler Schematic

Chapel Source Code © Chapel-to-C Compiler © Chapel Executable (a.out) © Chapel Standard Modules
Standard modules implement standard library routines.

- **BlockDist**: Definition of Block distribution
- **BitOps**: Specialized bit manipulation
- **Random**: Random number generation
- **Time**: Timer and time-of-day support

Standard modules must be explicitly used

E.g., `use BlockDist;`
Chapel: Compiler Overview

Compiler Schematic

- Chapel Source Code
- Chapel-to-C Compiler
- Chapel Standard Modules
- Chapel Executable (a.out)
Chapel: Compiler Overview

Chapel-to-C Compiler

Chapel Source Code

Chapel-to-C Compiler

Chapel Standard Modules

Chapel-to-C Compiler

Generated C Code

Standard C Compiler & Linker

Chapel Executable (a.out)

Internal Modules (Written in Chapel)

Runtime Support Libraries (Written in C)

1-Sided Communication and Threading Libraries (E.g., GASNet and pThreads)
Internal modules implement basic Chapel features.

- Standard operators
- Standard math routines
- User-level I/O routines
- User-level assertions and halts
- Tuples, ranges, domains, and arrays
- Synchronization variables

Essential to development

- Improves robustness by using the language
- Makes development easier because Chapel is productive
Chapel Source Code → Chapel-to-C Compiler → Generated C Code → Standard C Compiler & Linker → Chapel Executable (a.out)

- Chapel-to-C Compiler
- Internal Modules (Written in Chapel)
- Runtime Support Libraries (Written in C)
- 1-Sided Communication and Threading Libraries (E.g., GASNet and pThreads)
Runtime Support Libraries

Runtime support libraries bootstrap Chapel.

- Command-line argument passing
- Console and file I/O primitives
- Error handling
- Memory management
- Type conversions
- Time primitives
- Thread creation and management
- Inter-process communication and coordination

This functionality has been migrating to Chapel.
Outline

- Chapel Compiler System Overview
- Version 0.9 Release and Status
Chapel Version 0.9

- Available on SourceForge
- Distributed BSD Open Source license
- Systems: Linux/Unix, Mac, Cygwin
- Contents
  - Compiler and standard modules
  - Runtime and third-party libraries (e.g., GASNet)
  - Top-level README for quick start
  - Language spec, quick reference, HPCC tutorial
  - Examples (tutorials, programs, and HPCC benchmarks)
  - Portability scripts
Implementation Status

- Base language and task parallelism
  - Complete with minor gaps (e.g., multiple inheritance)

- Data parallelism
  - Serial reference implementation
  - Initial support for concurrency via distributions

- Distributed memory
  - Task parallelism across locales
  - Initial support for distributed arrays and domains

- Performance
  - Focus on a small set of language features
Unimplemented Features Seen Today

- **Base language**
  - Constness is not checked for domains, arrays, fields

- **Task parallelism**
  - Atomic statements are not atomic

- **Data parallelism**
  - Promoted functions/operators do not preserve shape
  - Reductions and scans cannot be user-defined or partial
  - Arrays of arrays require inner arrays to use a single domain

- **Locality and affinity**
  - User-defined distributions are not yet specified