Chapel

Introduction

- Chapel: Cascade High Productivity Language
- Developed by Cray as part of DARPA's HPCS program, with collaborations from academia
- Motivating Themes:
  - PGAS memory model: to specify & manage locality
  - general parallelism: data, task, and nested
  - modern language features:
    - OOP
    - generic programming, latent types
    - generators

Overview

A dynamically varying number of tasks…

- created with task parallel concepts: begins, coforalls, …
- or data parallel concepts: ZPL-like domains and arrays

…operating on a fixed number of *locales*…

- an architectural unit of memory locality + processing
- think multicore processor + memory, or SMP node

…synchronizing through memory

- synchronization variables with full/empty state & value
- plus software transactional memory concepts (STM)

Chapel and Productivity

- programmability
  - simpler to write parallel codes
  - and also to read, modify, port, tune, maintain them
- performance
  - competitive with MPI on generic clusters
  - better than MPI on more advanced architectures
- portability
  - as ubiquitous as MPI, but with fewer assumptions
- code robustness
  - better abstractions to minimize common errors

Resources

- Chapel Home Page
  - http://chapel.cs.washington.edu
- Recommended Starting Points
  - *Parallel Programmability and the Chapel Language,* IJHPCA, August 2007
  - HPCC Benchmarks in Chapel: Overview & Tutorial
  - Chapel Language Specification 0.750
- Contact Address:
  - chapel_info@cray.com