Chapel Overview

Greg Titus, Chapel Team, Cray Inc.

Chapel Lightning Talks

November 18th, 2014

SC14
New Orleans, LA | hpc matters.

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

What is Chapel?

- **An emerging parallel programming language**
  - Design and development led by Cray Inc.
    - with contributions from academics, labs, industry
  - Initiated under the DARPA HPCS program

- **Overall goal: Improve programmer productivity**

- **A work-in-progress**

Chapel's Implementation

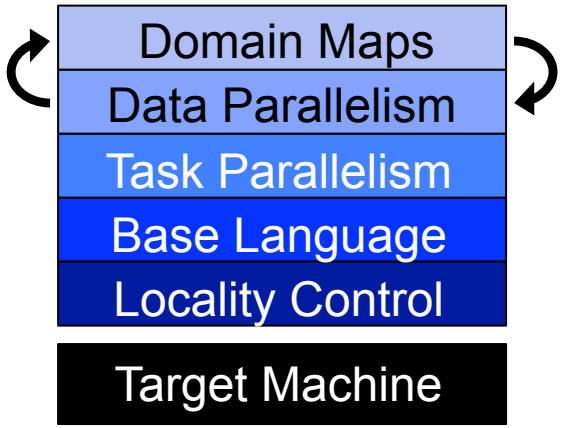
- **Being developed as open source at GitHub**
- **Licensed as Apache software**
- **Target Architectures:**
  - Cray architectures
  - multicore desktops and laptops
  - commodity clusters
  - systems from other vendors
  - (in-progress: CPU+accelerator hybrids, manycore, …)

Multiresolution Design: a Fundamental Concept

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

Chapel in a Nutshell: Task Parallelism, etc.

Variables and types for reasoning about system resources:
Locales: the collection of compute nodes on which the program is running
here: the node on which the current task is running

Syntactic constructs for creating task parallelism:
coforall (concurrent forall): creates a task per iteration

taskParallel.chpl

```
coforall loc in Locales do
  on loc {
    const numTasks = here.maxTaskPar;
    coforall tid in 1..numTasks do
      writef("Hello from task %n of %n running on %s\n",
             tid, numTasks, here.name);
  }
```

Control over locality/affinity:
on-clauses: task migration

Static type inference (optionally):
Supports programmability with performance

```
prompt> chpl taskParallel.chpl -o taskParallel
prompt> ./taskParallel --numLocales=2
Hello from task 1 of 4 running on n1032
Hello from task 4 of 4 running on n1032
Hello from task 2 of 4 running on n1033
Hello from task 1 of 4 running on n1033
Hello from task 3 of 4 running on n1032
Hello from task 3 of 4 running on n1033
Hello from task 2 of 4 running on n1032
Hello from task 4 of 4 running on n1033
```

Chapel in a Nutshell: Data Parallelism, etc.

Modules for namespace management:
CyclicDist: standard module providing cyclic distributions

Configuration variables and constants:
Never write an argument parser again (unless you want to)

Domains and Arrays:
Index sets and arrays that can optionally be distributed

Data parallel forall loops and operations:
Use available parallelism for data-driven computations

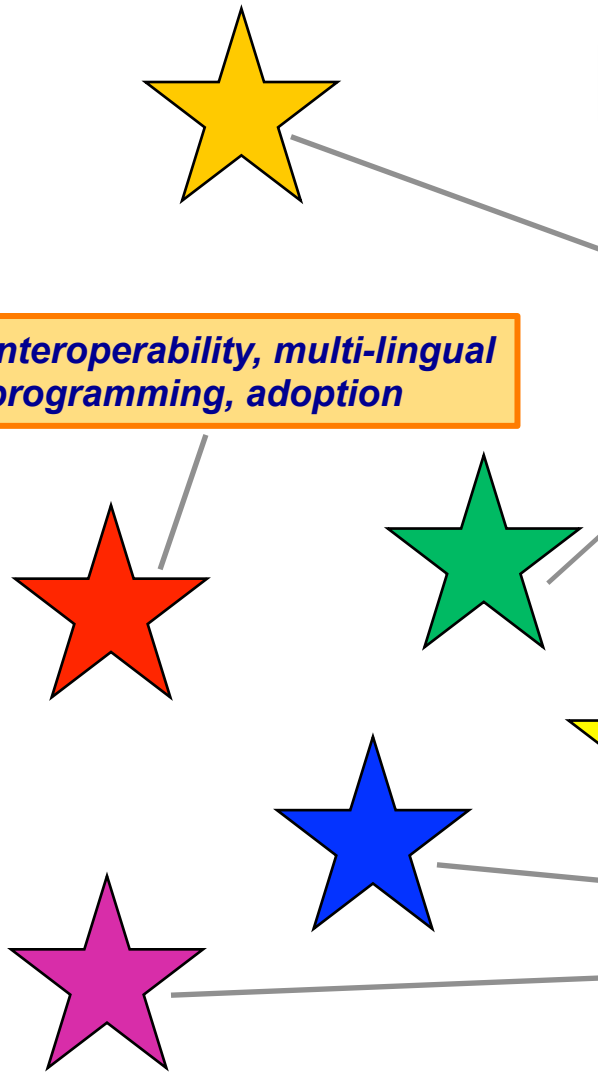
Domain maps
Describe how iterations over domains/arrays are mapped to locales

dataParallel.chpl

```
use CyclicDist;
config const n = 1000;
var D = {1..n, 1..n}
        dmapped Cyclic(startIdx = (1,1));
var A: [D] real;
forall (i,j) in D do
  A[i,j] = i + (j - 0.5)/n;
writeln(A);
```

```
prompt> chpl dataParallel.chpl -o dataParallel
prompt> ./dataParallel --numLocales=4 --n=5
1.1 1.3 1.5 1.7 1.9
2.1 2.3 2.5 2.7 2.9
3.1 3.3 3.5 3.7 3.9
4.1 4.3 4.5 4.7 4.9
5.1 5.3 5.5 5.7 5.9
```

Where Will Today's Talks Take Us?



taskParallel.chpl

```
coforall loc in Locales do
  on loc {
    const numTasks = here.maxTaskPar;
    coforall tid in 1..numTasks do
      writef("Hello from task %n of %n running on %s\n",
             tid, numTasks, here.name);
  }
```

Interoperability, multi-lingual programming, adoption

dataParallel.chpl

```
use CyclicDist;
config const n = 1000;
var D = {1..n, 1..n}
      dmapped Cyclic(startIdx = (1,1));
var A: [D] real;
forall (i,j) in D do
  A[i,j] = i + (j - 0.5)/n;
writeln(A);
```

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.