Welcome, Introduction to Chapel, State of the Project

Brad Chamberlain, Chapel Team, Cray Inc.

CHIUW: Chapel Implementers and Users Workshop

May 23rd, 2014

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.
    - in collaboration with academia, labs, industry
- **Overall goal: Improve programmer productivity**
  - Improve the programmability of parallel computers
  - Match or beat the performance of current programming models
  - Support better portability than current programming models
  - Improve the robustness of parallel codes
- **A work-in-progress**

Chapel's Implementation

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

A Year in the Life of Chapel

- **Two major releases per year** (April / October)
  - **latest release:** version 1.9, April 17, 2014
  - **a month later:** detailed release notes (new!)
    - version 1.9 release notes available now: http://chapel.cray.com/download.html
- **SC activities** (Nov)
  - **Chapel tutorials** (most years)
  - **CHUG meet-up / happy hour** (past four years)
  - **lightning talks BoF** (past three years)
  - **educators forum** (past two years)
  - **talks, posters, emerging technology booth, etc.** (when possible)
- **Talks, panels, tutorials, research visits, papers, blog articles, …** (year-round)
- and now… **CHIUW** (May)

Welcome to CHIUW!

What is CHIUW?

CHIUW: Chapel Implementers and Users Workshop

- Name chosen to complement CHUG (the Chapel Users Group)

Motivation:

- Lots of things going on with Chapel
  - within Cray
  - across broader research community
- Wanted to create a forum to share progress and discuss

Format:

- backbone: technical talks submitted from Chapel community
- networking during breaks and meals
- fleshing things out: invited keynote talk, community/panel discussion

CHIUW 2014 Agenda

8:30– 9:00	Welcome, Overview
9:00–10:00	Technical Talks: **Application Studies**
10:00–10:30	Break
10:30–11:30	Technical Talks: **Language Extensions**
11:30–12:00	Technical Talks: **Compiler Optimizations (pt 1)**
12:00– 1:00	Lunch
1:00– 1:45	Keynote: **Walking to the Chapel, Robert Harrison**
1:45– 2:15	Technical Talks: **Compiler Optimizations (pt 2)**
2:15– 3:15	Technical Talks: **Runtime Improvements**
3:15– 3:30	Break
3:30– 4:30	**Community/Panel Discussion**
4:30–	Dinner/Drinks

(Full agenda at http://chapel.cray.com/CHIUW.html)

CHIUW 2014 Talks and Speakers

(Full author lists and extended abstracts at http://chapel.cray.com/CHIUW.html)

User Experiences with a Chapel Implementation of UTS
Jens Breitbart, Technische Universität München

Evaluating Next Generation PGAS Languages for Computational Chemistry
Daniel Chavarria-Miranda, Pacific Northwest National Laboratory

Programmer-Guided Reliability in Chapel
David E. Bernholdt, Oak Ridge National Laboratory

Towards Interfaces for Chapel
Chris Wailes, Indiana University

Affine Loop Optimization using Modulo Unrolling in Chapel
Aroon Sharma, University of Maryland

LLVM Optimizations for PGAS Programs
Akihiro Hayashi, Rice University

Opportunities for Integrating Tasking and Communication Layers
Dylan T. Stark, Sandia National Laboratories

Caching in on Aggregation
Michael Ferguson, Laboratory for Telecommunication Sciences

CHIUW 2014 Steering Committee

- **Brad Chamberlain (chair),** Cray Inc.
- **Richard Barrett,** Sandia National Laboratories
- **Jens Breitbart,** Technische Universität München
- **Michael Ferguson,** Laboratory for Telecommunication Sciences
- **Rob Neely,** Lawrence Livermore National Laboratory
- **Vivek Sarkar,** Rice University
- **Jeremy Siek,** Indiana University
- **Kenjiro Taura,** University of Tokyo

In forming the committee, strived for a balance of…

- developers and users
- academics and lab employees
- domestic and international

The Future of CHIUW

- **Intention is to make this an annual event**
- **Format and setting are flexible based on interest**
  - don't change a thing?
  - same format, different activities?
  - multiple days with distinct user/implementer emphases?
  - coding camps for developers/users?
  - tutorials?
- **Sometime today, please fill out survey to help tune CHIUW**
- **Interested in chairing CHIUW 2015? Please let us know!**

Outline For This Talk

- ✓ Welcome / Context
- ➢ Chapel Overview
- ● Chapel Current Events

Chapel Overview

Chapel's Origins: HPCS

DARPA HPCS: High Productivity Computing Systems

- **Goal:** improve productivity by a factor of 10x
- Timeframe: summer 2002 – fall 2012
- Cray developed new system architecture, network, software, …
  - this became the very successful Cray XC30™ Supercomputer Series

…and a new programming language: Chapel

Major Chapel Successes Under HPCS

- **SSCA#2 demonstration on the prototype Cray XC30**
  - unstructured graph compact application
  - clean separation of computation from data structure choices
  - fine-grain latency-hiding runtime
  - use of XC30's network AMOs via Chapel's 'atomic' types

- **Clean, general parallel language design**
  - unified data-, task-, concurrent-, nested-parallelism
  - distinct concepts for parallelism and locality
  - multiresolution language design philosophy

- **Portable design and implementation**
  - while still being able to take advantage of Cray-specific features

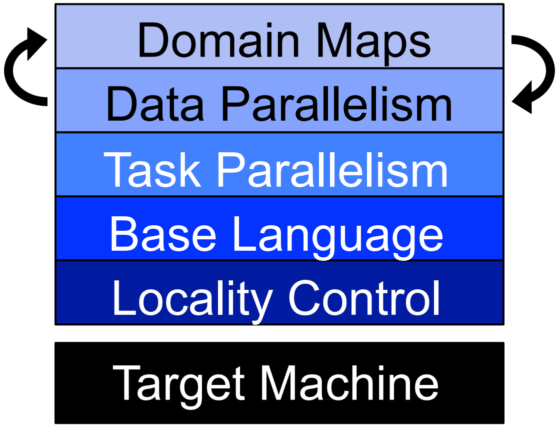
- **Revitalization of Community Interest in Parallel Languages**
  - HPF-disenchantment became interest, cautious optimism, enthusiasm

Multiresolution Design

Multiresolution Design: **Support multiple tiers of features**

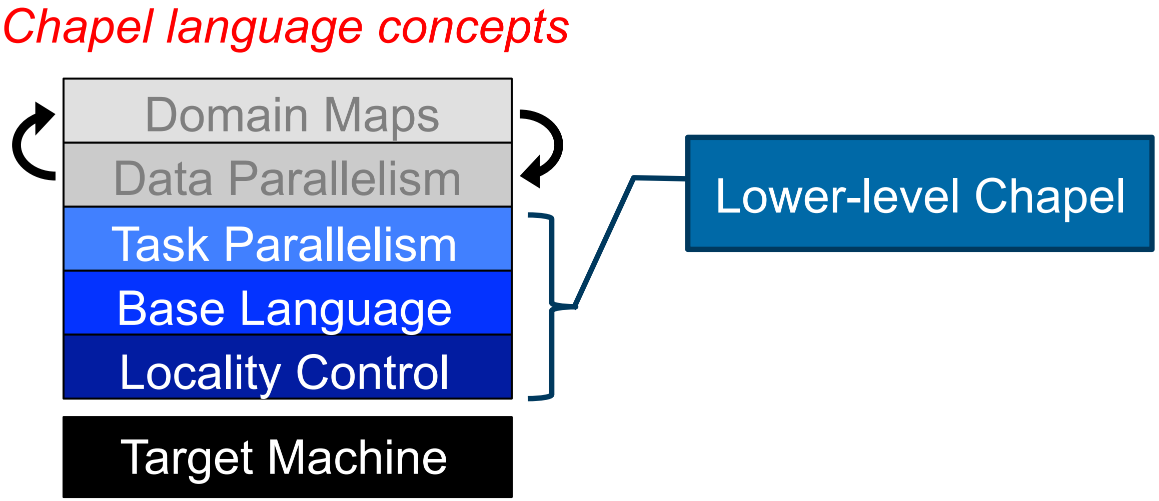
- higher levels for programmability, productivity
- lower levels for greater degrees of control

Chapel language concepts

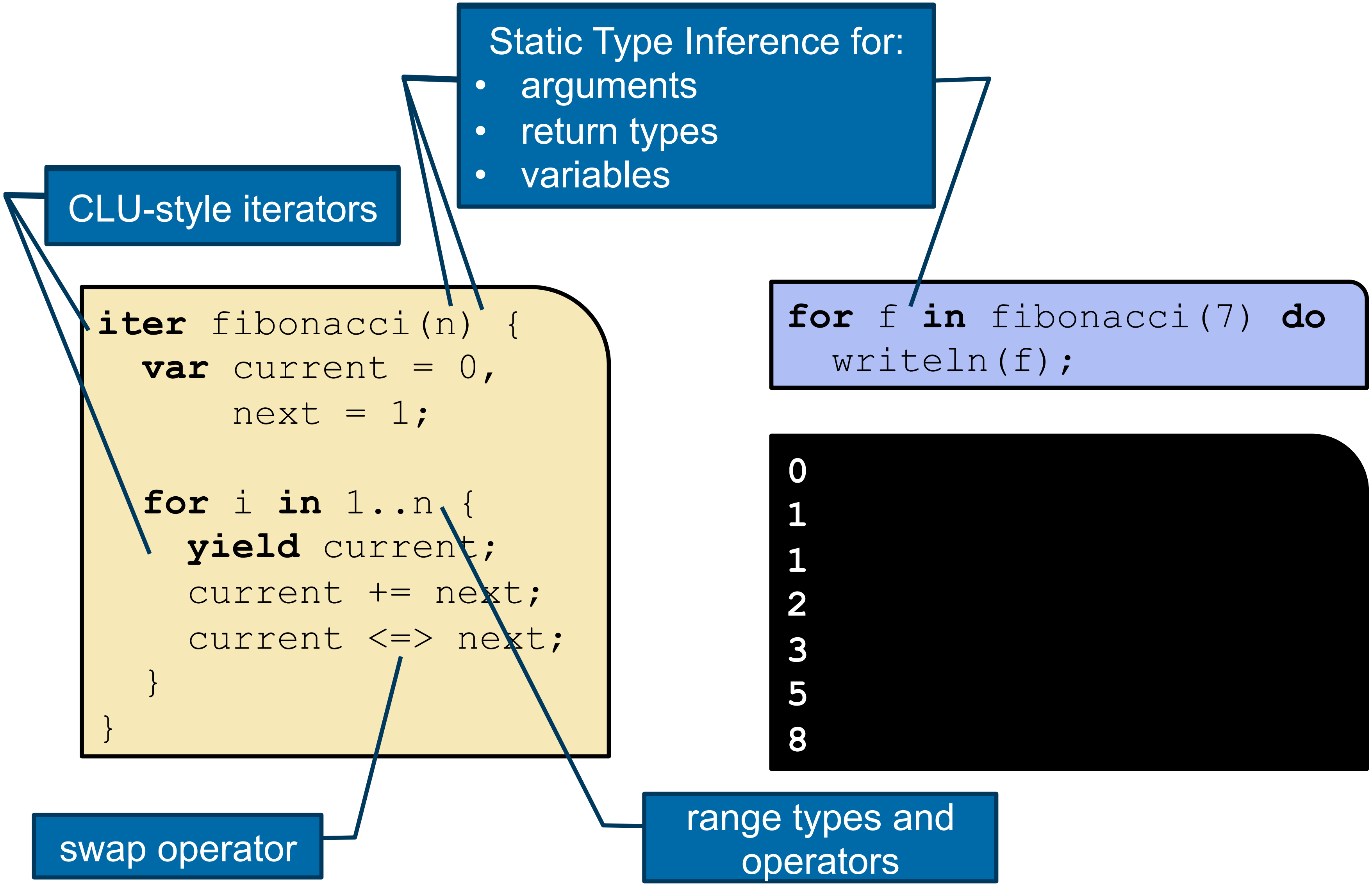


- build the higher-level concepts in terms of the lower
- permit the user to intermix layers arbitrarily

Lower-Level Features



Sample Base Language Features



Static Type Inference for:

- arguments
- return types
- variables

CLU-style iterators

```
iter fibonacci(n) {
  var current = 0,
      next = 1;

  for i in 1..n {
    yield current;
    current += next;
    current <=> next;
  }
}
```

```
for f in fibonacci(7) do
  writeln(f);
```

```
0
1
1
2
3
5
8
```

swap operator

range types and operators

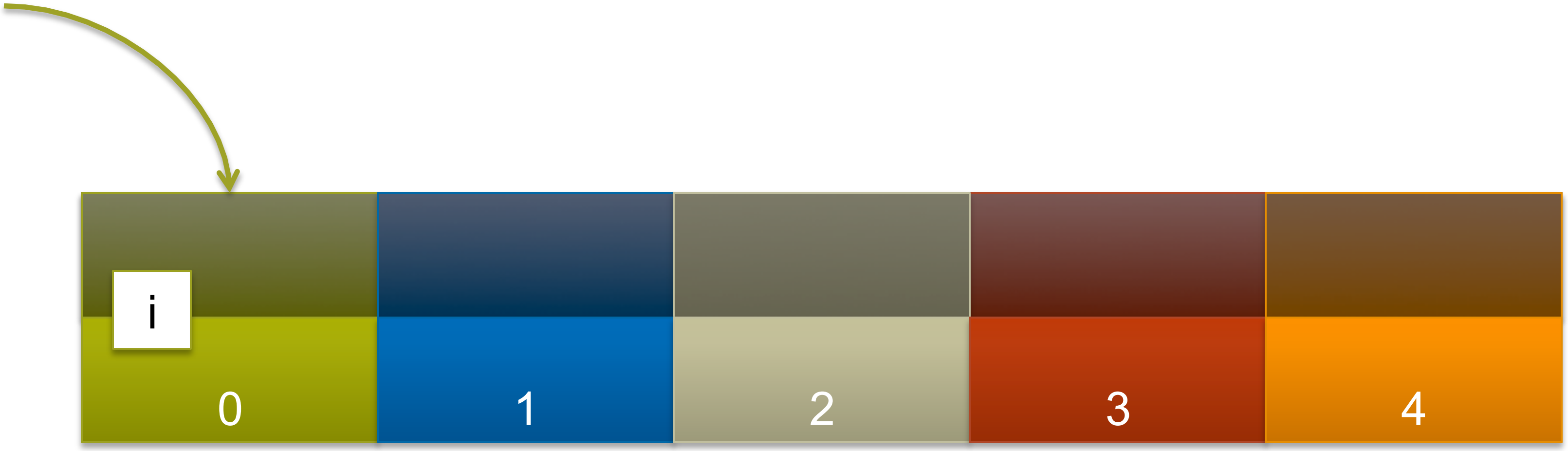
Sample Task Parallel Feature: Coforall Loops

```
coforall t in 0..#numTasks {   // coforalls create a task per iteration
  writeln(“Hello from task ”, t, “ of ”, numTasks);
}   // its tasks implicitly join before execution proceeds

writeln(“All tasks done”);
```

```
Hello from task 2 of 4
Hello from task 0 of 4
Hello from task 3 of 4
Hello from task 1 of 4
All tasks done
```

Locality Features: Locales and on-clauses

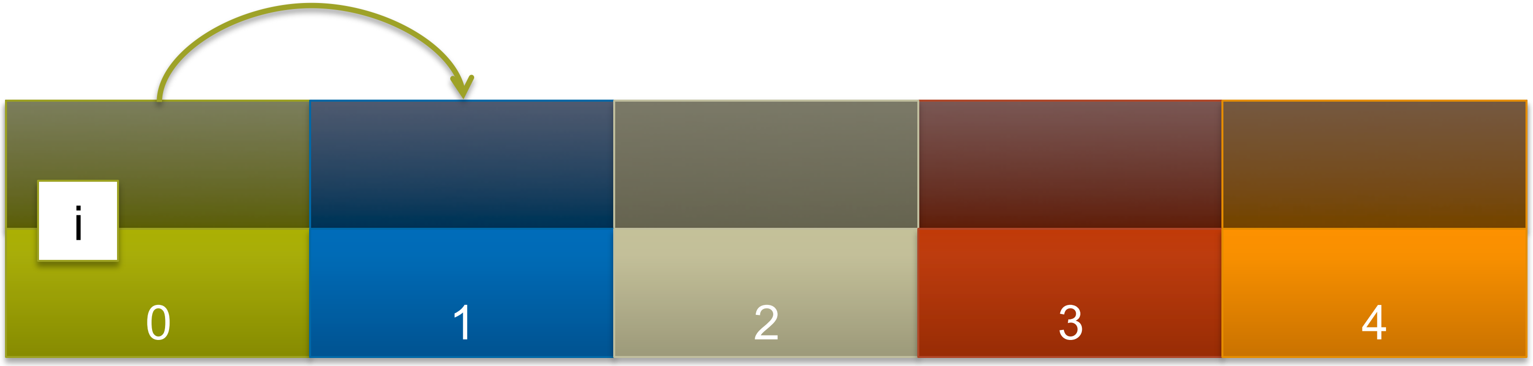
```
var i: int;
```



Locales (think: “compute nodes”)

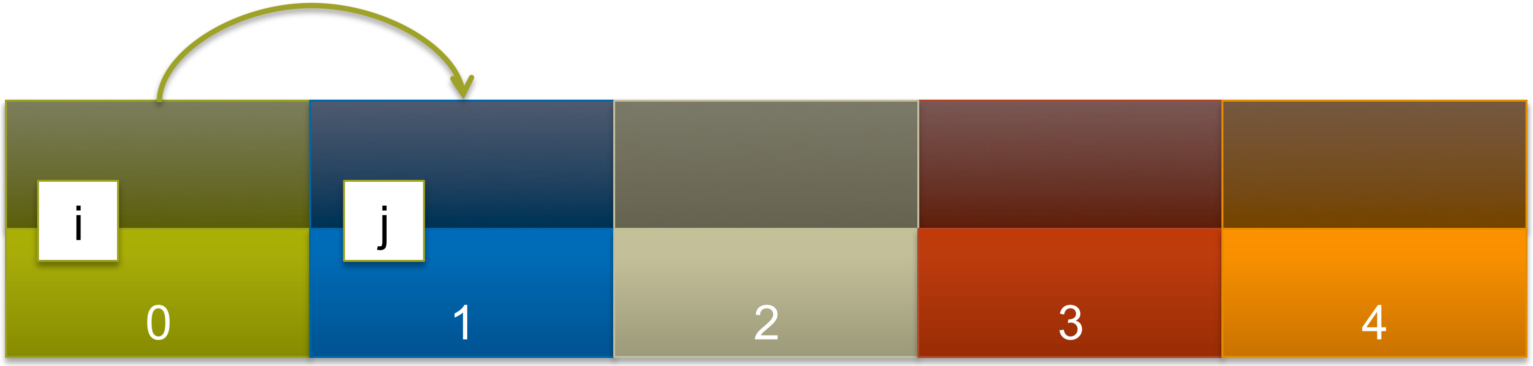
Locality Features: Locales and on-clauses

```
var i: int;
on Locales[1] {
```



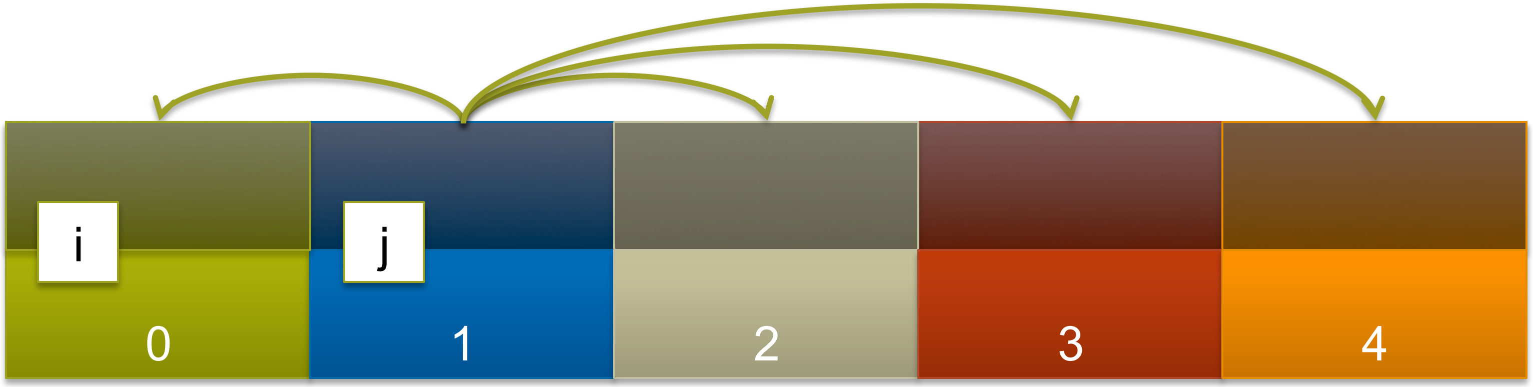
Locality Features: Locales and on-clauses

```
var i: int;
on Locales[1] {
  var j: int;
```



Locality Features: Locales and on-clauses

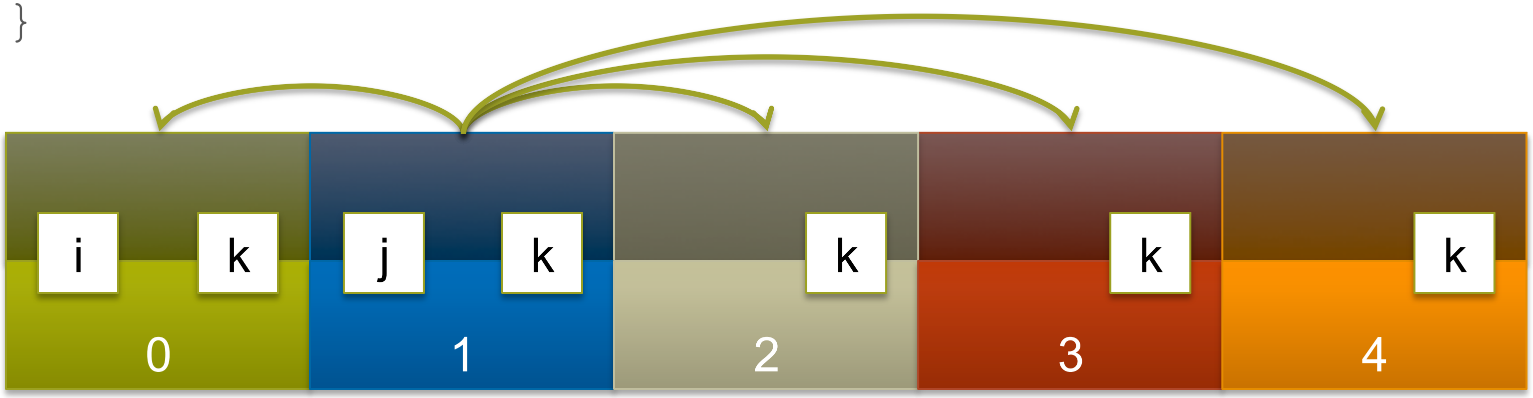
```
var i: int;
on Locales[1] {
  var j: int;
  coforall loc in Locales {
    on loc {
```



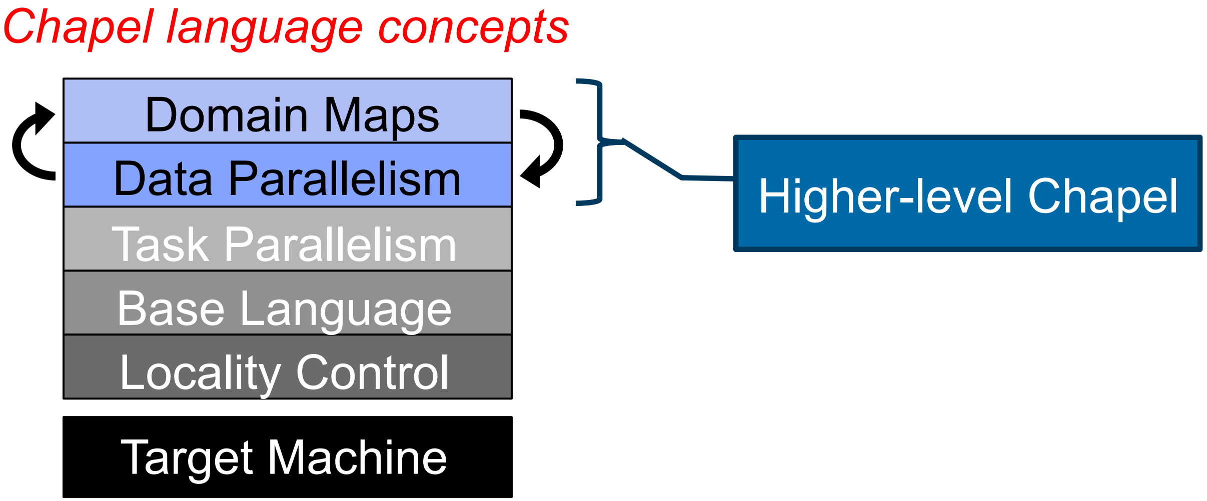
Copyright 2014 Cray Inc.

Locality Features: Locales and on-clauses

```
var i: int;
on Locales[1] {
  var j: int;
  coforall loc in Locales {
    on loc {
      var k: int;

      // within this scope, i, j, and k can be referenced;
      // the implementation manages the communication for i and j
    }
  }
}
```

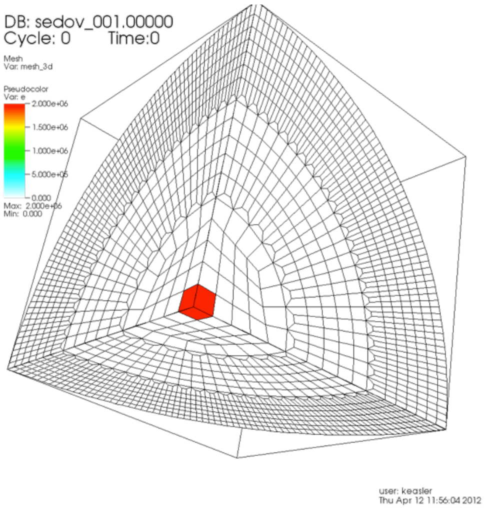


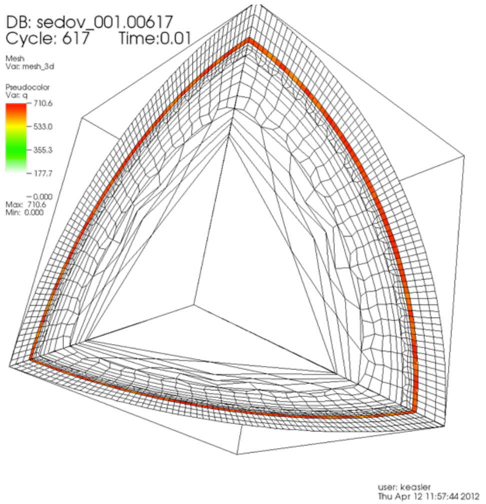
Higher-Level Features

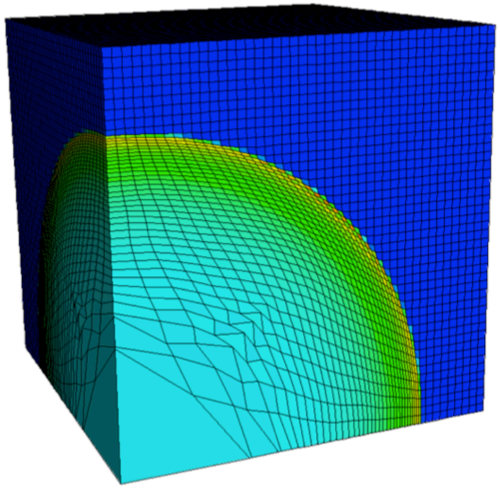


LULESH: a DOE Proxy Application

Goal: Solve one octant of the spherical Sedov problem (blast wave) using Lagrangian hydrodynamics for a single material





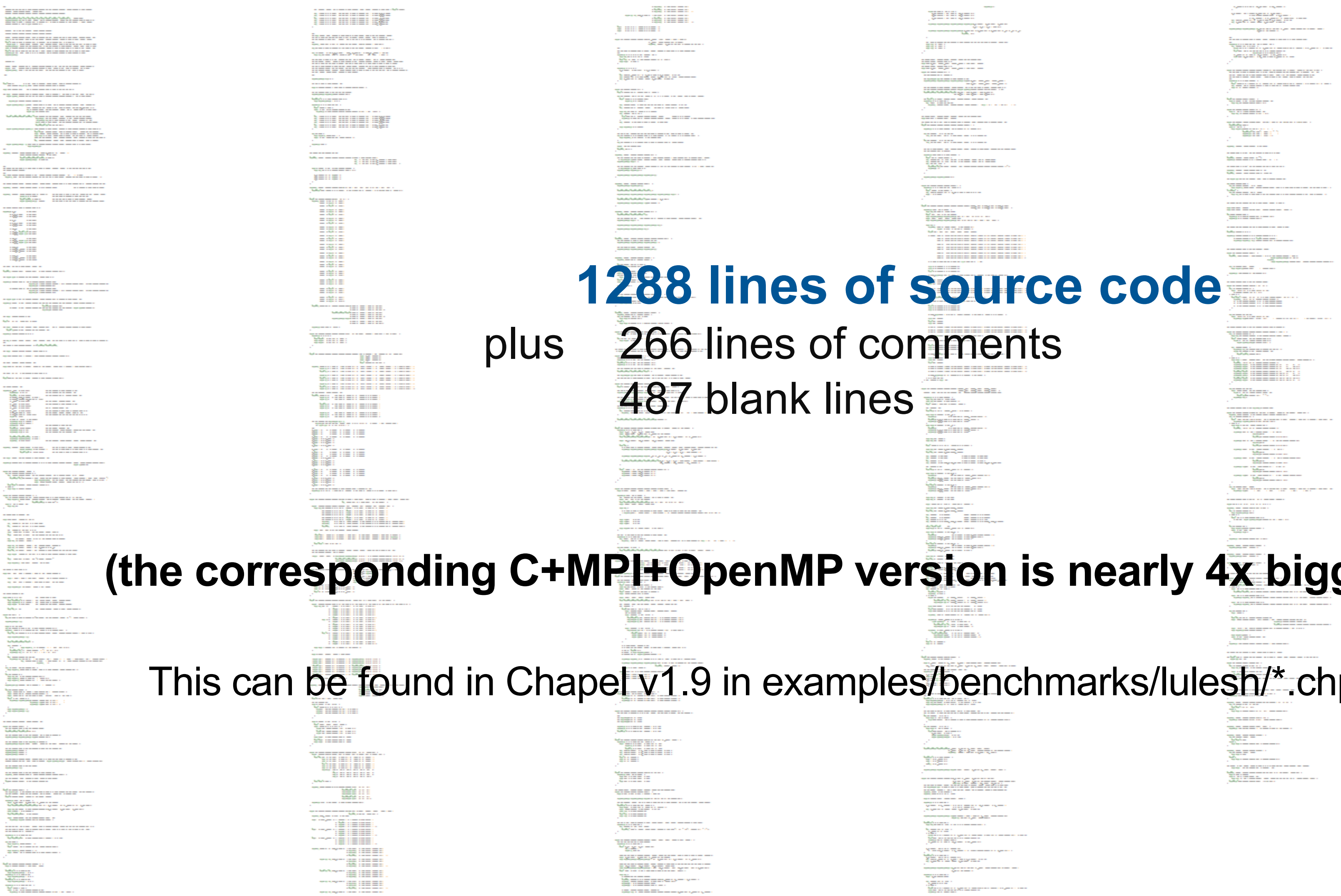


pictures courtesy of Rob Neely, Bert Still, Jeff Keasler, LLNL

LULESH in Chapel

[illegible]

LULESH in Chapel



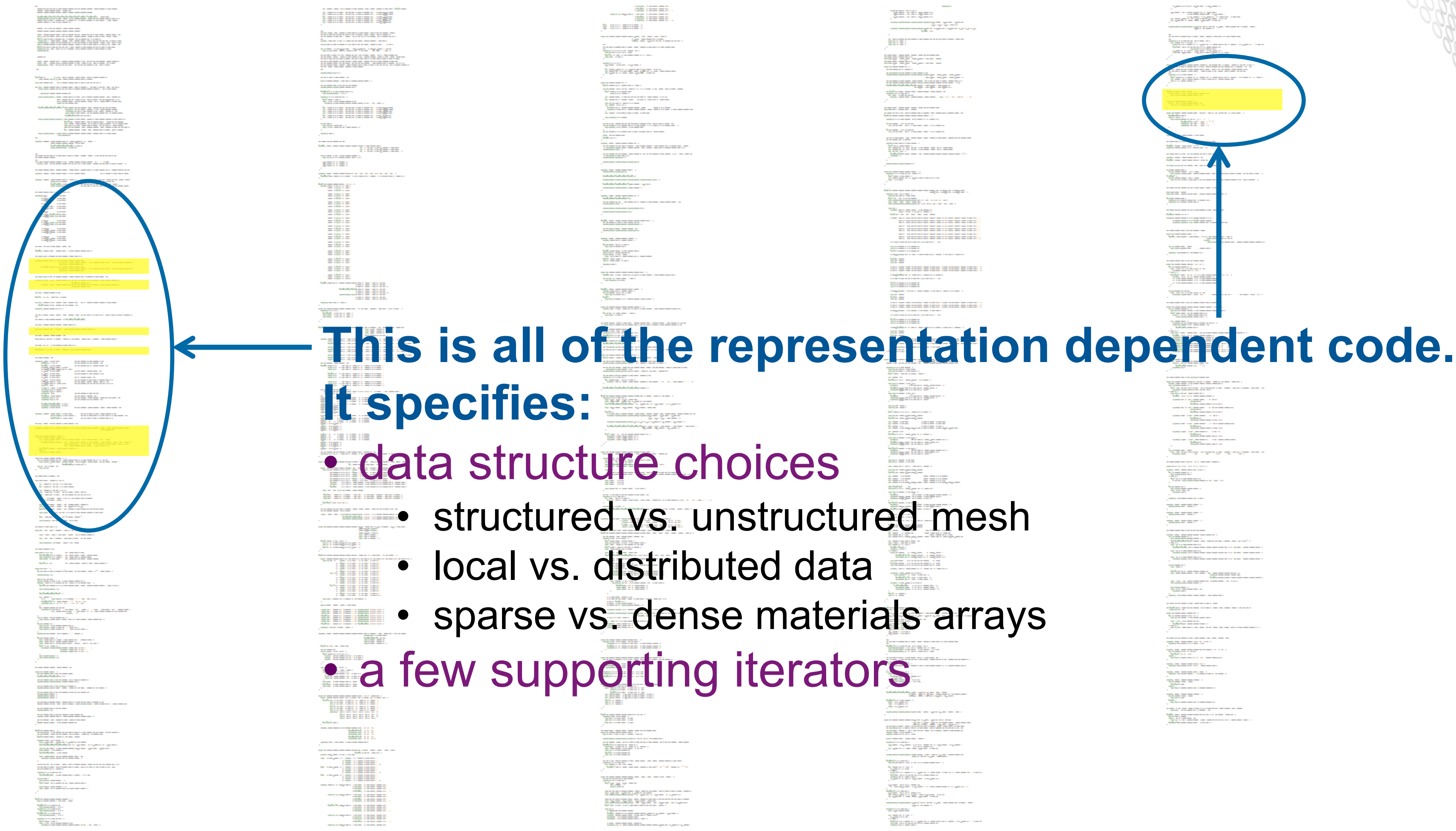
1288 lines of source code

plus 266 lines of comments
487 blank lines

(the corresponding C+MPI+OpenMP version is nearly 4x bigger)

This can be found in Chapel v1.9 in examples/benchmarks/lulesh/*.chpl

LULESH in Chapel

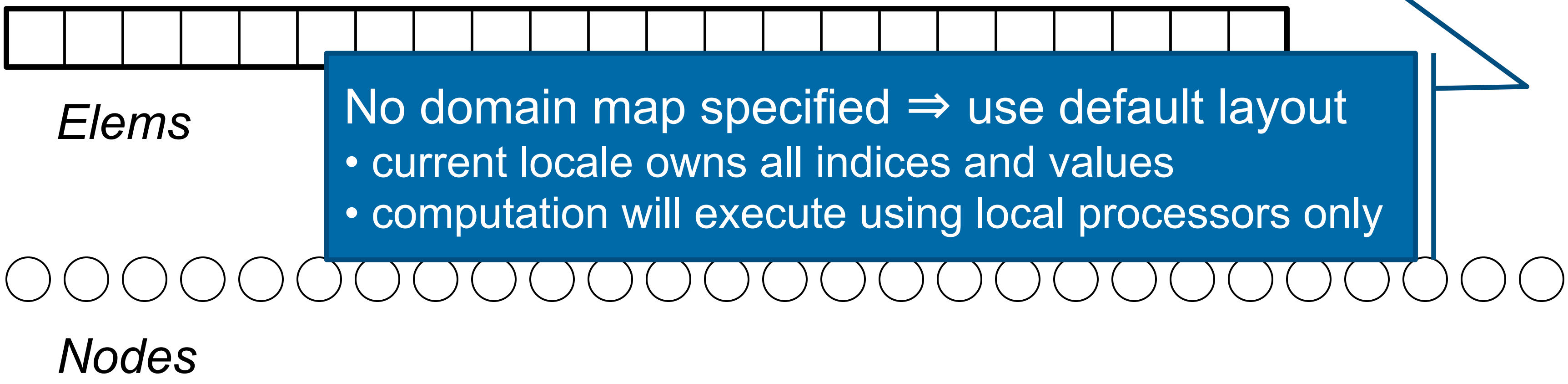


This is all of the representation dependent code.
It specifies:

- data structure choices
  - structured vs. unstructured mesh
  - local vs. distributed data
  - sparse vs. dense materials arrays
- a few supporting iterators

LULESH Data Structures (local)

```
const Elems = {0..#numElems},
      Nodes = {0..#numNodes};

var determ: [Elems] real;

forall k in Elems { … }
```

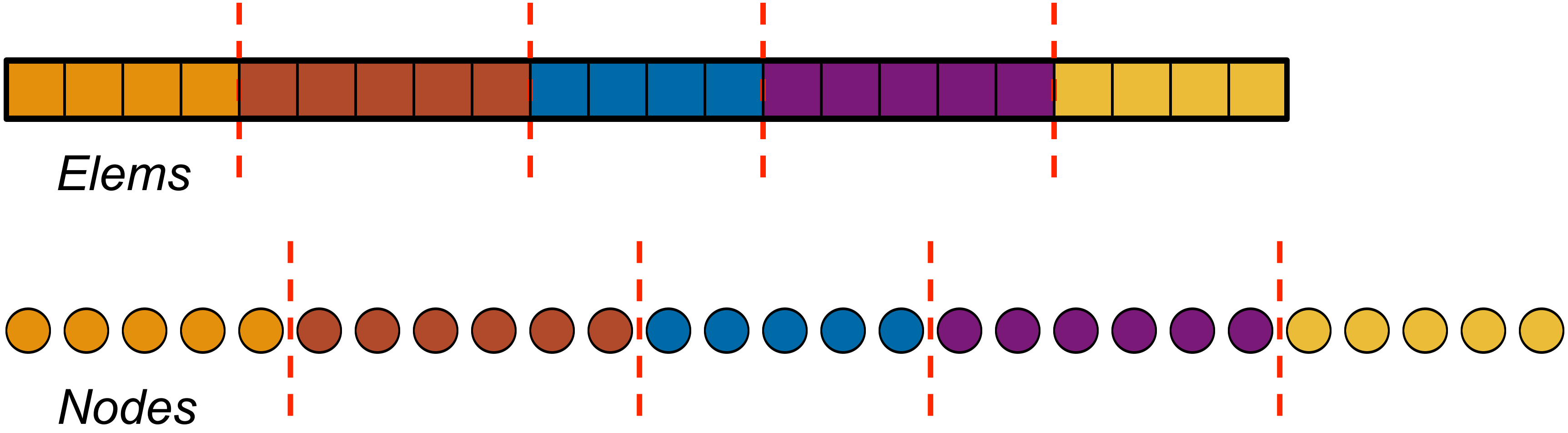


No domain map specified ⇒ use default layout

- current locale owns all indices and values
- computation will execute using local processors only

LULESH Data Structures (distributed, block)

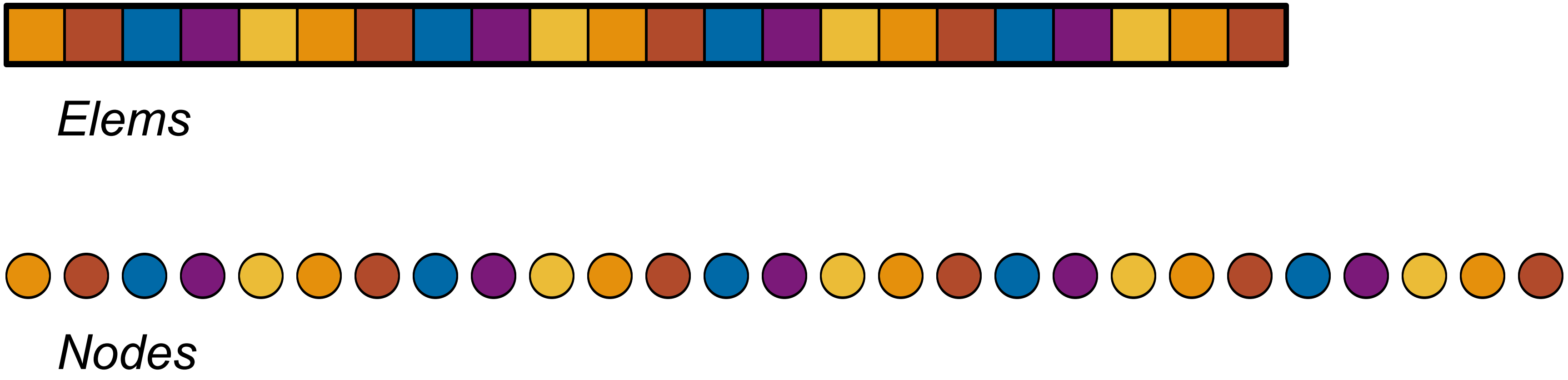
```
const Elems = {0..#numElems} dmapped Block(…),
      Nodes = {0..#numNodes} dmapped Block(…);

var determ: [Elems] real;

forall k in Elems { … }
```



Copyright 2014 Cray Inc.

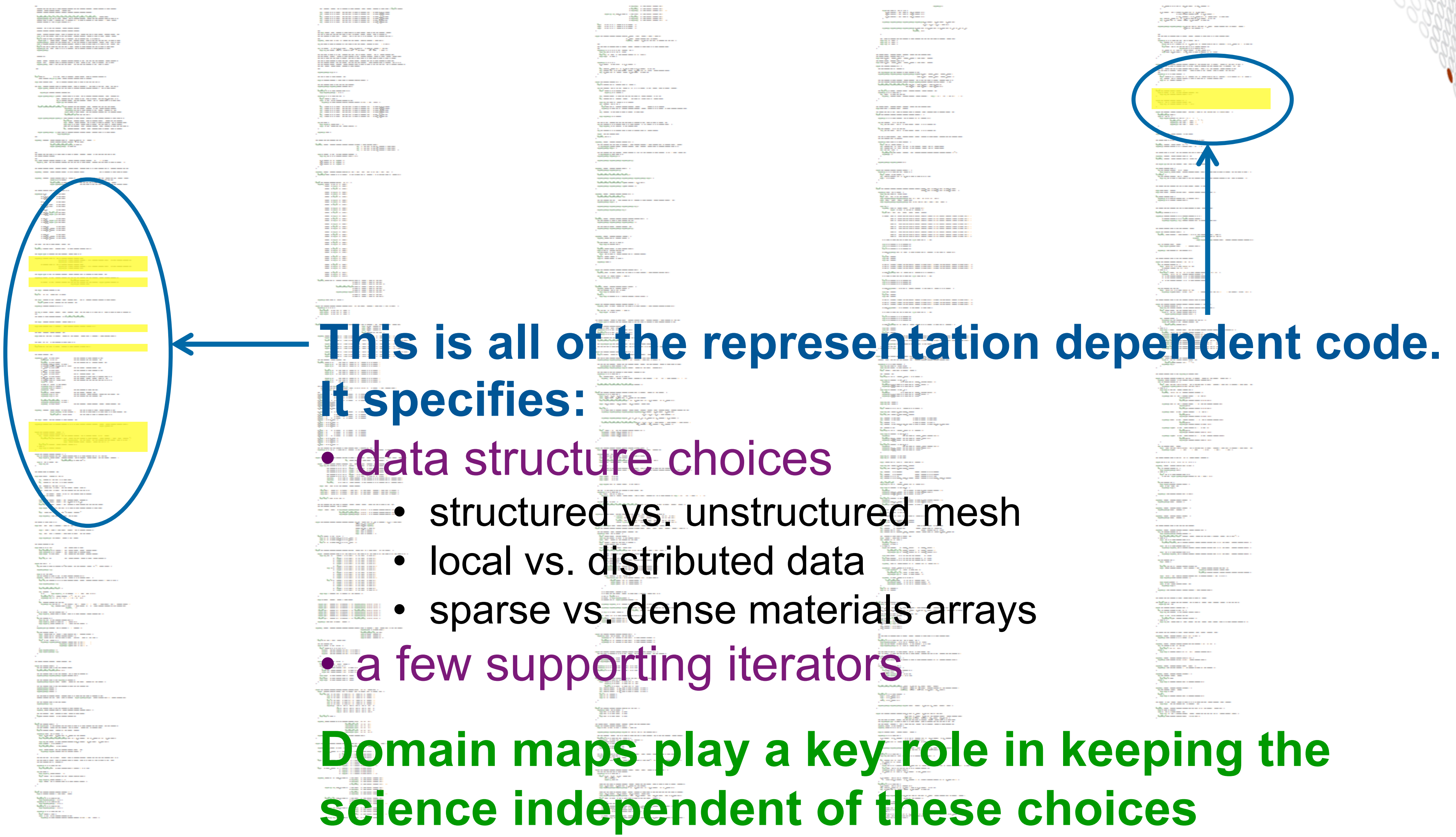
LULESH Data Structures (distributed, cyclic)

```
const Elems = {0..#numElems} dmapped Cyclic(…),
      Nodes = {0..#numNodes} dmapped Cyclic(…);

var determ: [Elems] real;

forall k in Elems { … }
```



Copyright 2014 Cray Inc.

LULESH in Chapel



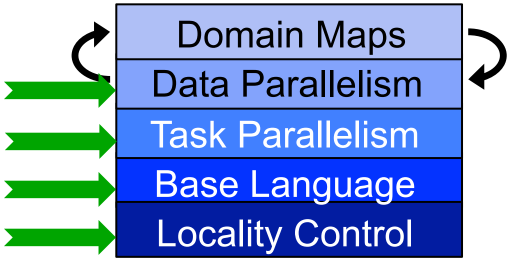
This is all of the representation dependent code. It specifies:

- data structure choices
  - structured vs. unstructured mesh
  - local vs. distributed data
  - sparse vs. dense materials arrays
- a few supporting iterators

Domain maps play a key role in keeping the science independent of these choices

Domain Maps and Multiresolution

- **Domain maps are written in Chapel**
  - by Chapel developers or end-users
  - using lower-level features



- **All domains/arrays are implemented using this framework**

For More Information: Suggested Reading

Overview Papers:

- *The State of the Chapel Union* [slides], Chamberlain, Choi, Dumler, Hildebrandt, Iten, Litvinov, Titus. CUG 2013, May 2013.
  - *a high-level overview of the project summarizing the HPCS period*

- *A Brief Overview of Chapel*, Chamberlain (pre-print of a chapter for *A Brief Overview of Parallel Programming Models*, edited by Pavan Balaji, to be published by MIT Press in 2014).
  - *a more detailed overview of Chapel's history, motivating themes, features*

Blog Articles:

- *[Ten] Myths About Scalable Programming Languages*, Chamberlain. IEEE Technical Committee on Scalable Computing (TCSC) Blog, (https://www.ieeetcsc.org/activities/blog/), April-November 2012.
  - *a series of technical opinion pieces designed to rebut standard arguments against the development of high-level parallel languages*

For More Information: Online Resources

Chapel project page: http://chapel.cray.com

- overview, papers, presentations, language spec, …

Chapel SourceForge page: https://sourceforge.net/projects/chapel/

- release downloads, public mailing lists, code repository, …

Mailing Lists:

- chapel_info@cray.com: contact the team at Cray
- chapel-announce@lists.sourceforge.net: announcement list
- chapel-users@lists.sourceforge.net: user-oriented discussion list
- chapel-developers@lists.sourceforge.net: developer discussion
- chapel-education@lists.sourceforge.net: educator discussion
- chapel-bugs@lists.sourceforge.net: public bug forum

Chapel Headlines

(“current events that every Chapel enthusiast should be aware of”)

Chapel is Alive and Well!

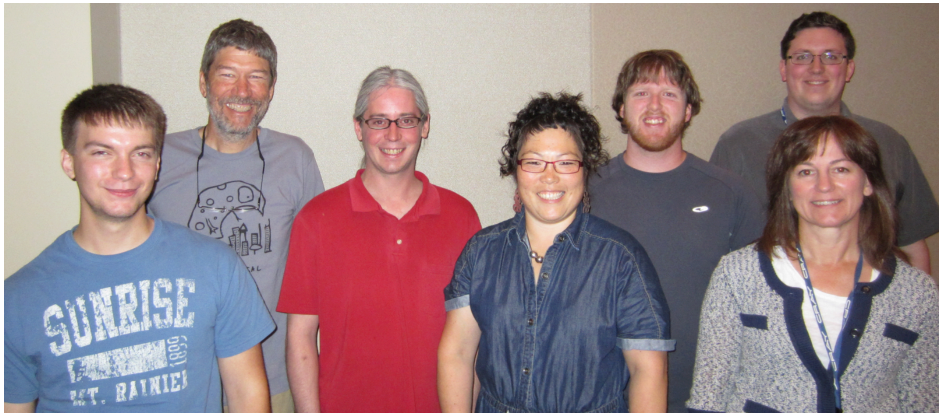
- **Based on positive customer response to Chapel under HPCS, Cray is pursuing a five-year effort to improve it**
  - the first year is drawing to a close

- **Focus Areas:**
  1. Performance and scaling improvements
  2. Fixing broken aspects of the language / implementation
     - e.g., error handling, strings, RAII/memory model, …
  3. Portability to modern-era architectures
     - Intel Phi, accelerators, heterogeneous memories, …
  4. Interoperability
  5. Community development
     - including non-HPC communities
  6. Transition governance to neutral, external group

The Chapel Team at Cray has Doubled in Size!

- **During HPCS, Chapel staffing was typically:**
  - 4-6 full-time engineers
  - a fraction of a manager's time

- **At present, the Cray Chapel team is composed of:**
  - 12 full-time engineers (6 here today)
  - a full-time manager (also here today)



Summer 2013

The Broader Chapel Community is also Growing!

RICE

LTS Laboratory for Telecommunications Sciences

ETH Zürich

Sandia National Laboratories

Lawrence Livermore National Laboratory

BERKELEY LAB
Lawrence Berkeley National Laboratory

Argonne NATIONAL LABORATORY

OAK RIDGE National Laboratory

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

東京大学
THE UNIVERSITY OF TOKYO

ma UNIVERSIDAD DE MÁLAGA

UNIVERSITY OF MARYLAND

(uptick in interest from industrial users/developers as well)

http://chapel.cray.com/collaborations.html

Everybody* Loves Chapel!

(but nobody will use it… yet)

Why is this?

1. lack of compelling performance demonstrations
   - Chapel performance is currently hit-or-miss
   - In distributed memory environments, typically “miss”

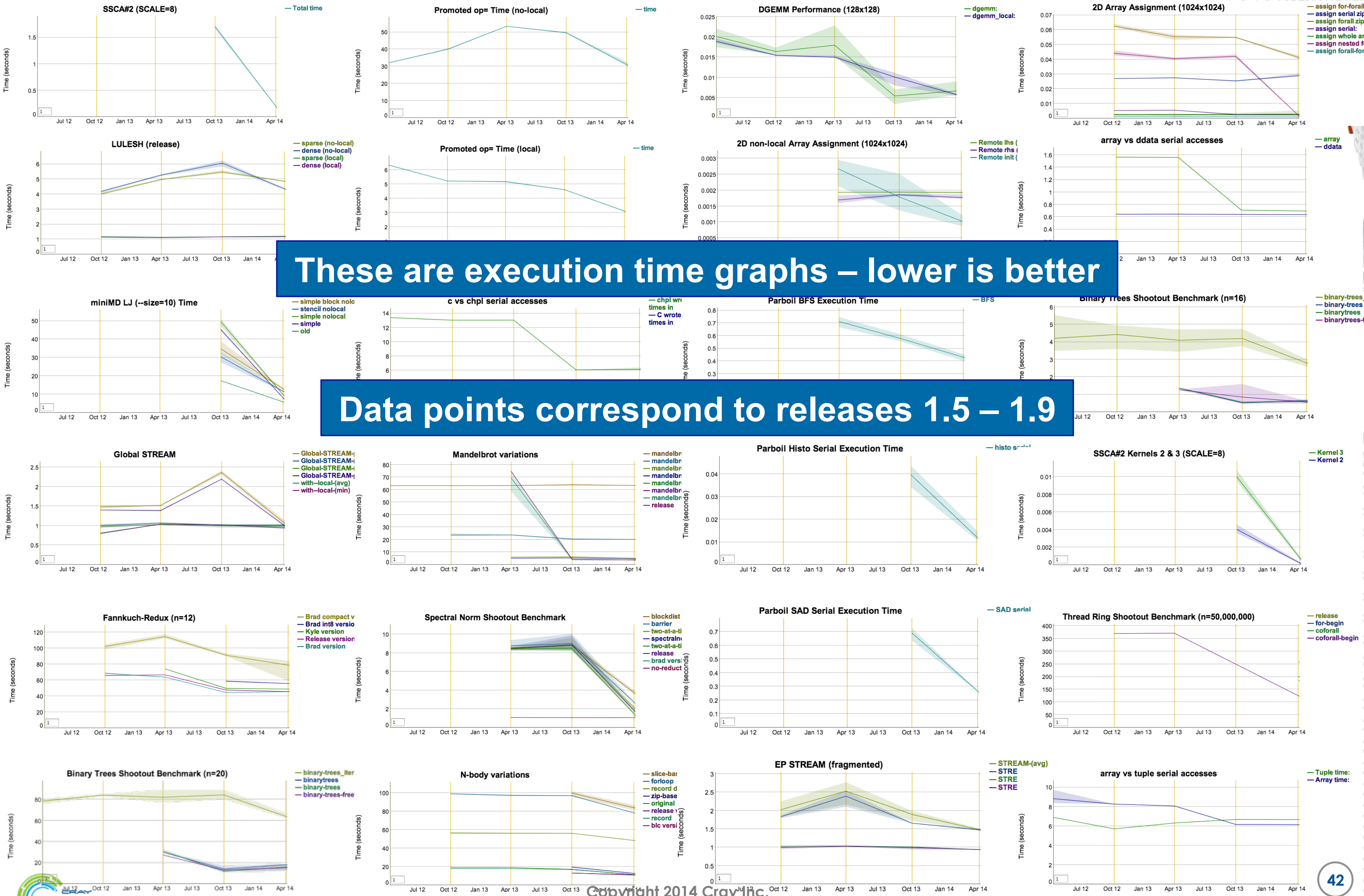
2. concern about long-term support / viability of a new language

3. lack of maturity in some areas (features, implementation)

4. insufficient interoperability features

* = well, OK, maybe just **most** people…

Performance is Improving!

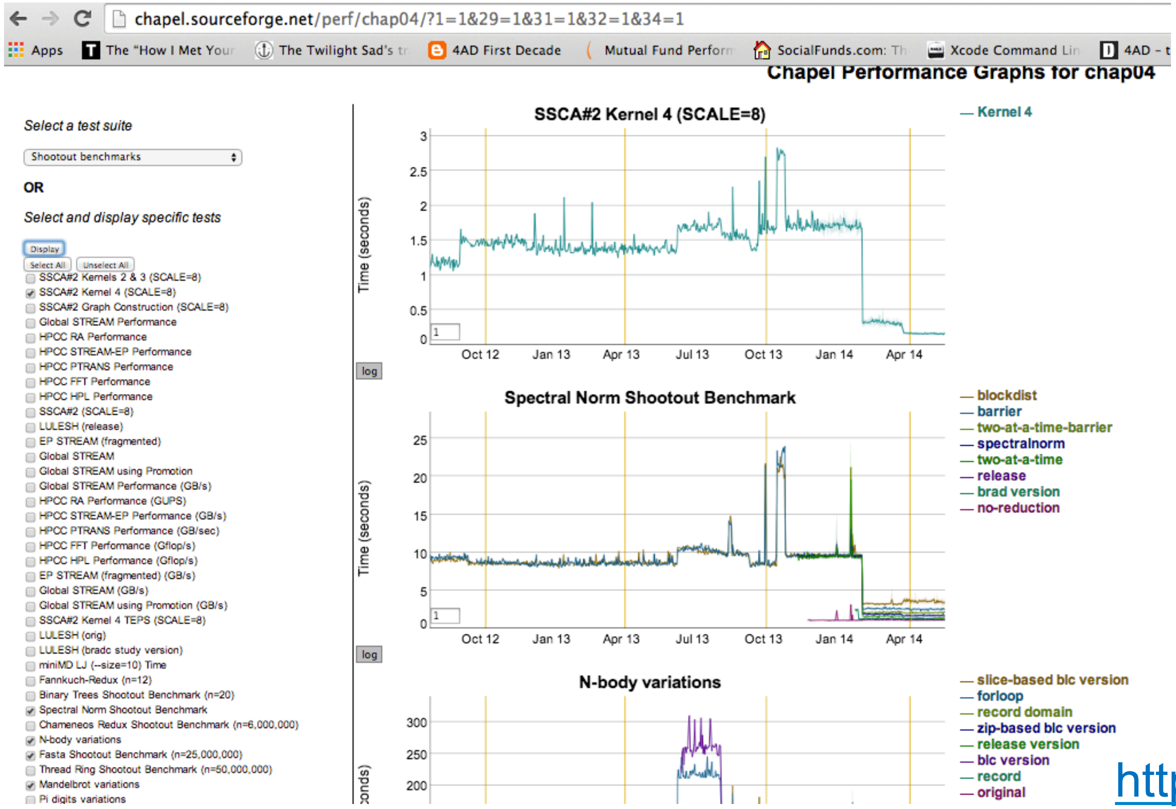


These are execution time graphs – lower is better

Data points correspond to releases 1.5 – 1.9

Nightly Performance Graphs are Now Public!

- **What this means:**
  - You can stalk our performance changes over time
  - You can submit your own performance tests and monitor them
  - You can see the performance impacts of patches you commit



chapel.sourceforge.net/perf/chap04/?1=1&29=1&31=1&32=1&34=1

Chapel Performance Graphs for chap04

Select a test suite

Shootout benchmarks

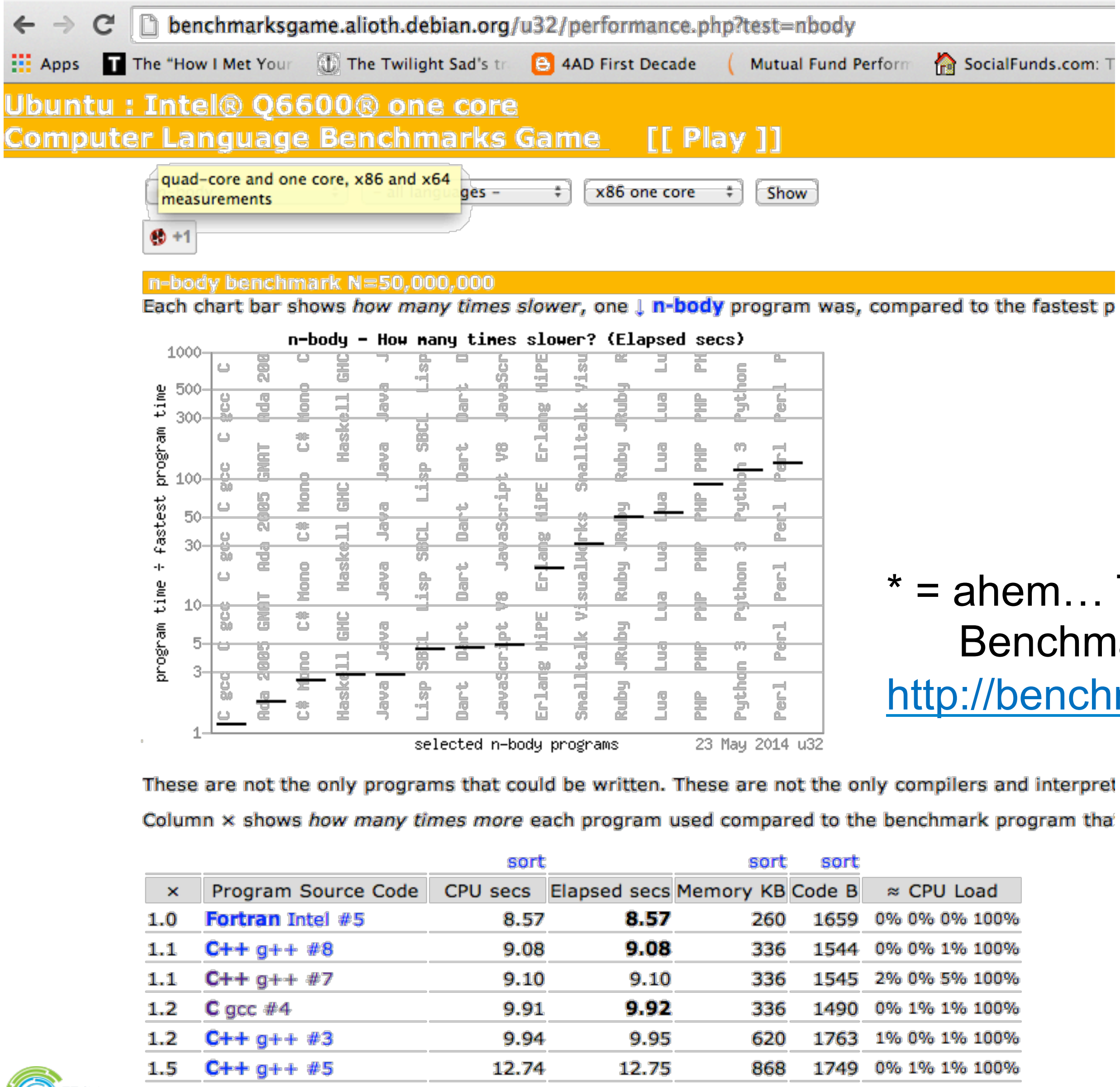
OR

Select and display specific tests

- □ SSCA#2 Kernels 2 & 3 (SCALE=8)
- ☑ SSCA#2 Kernel 4 (SCALE=8)
- □ SSCA#2 Graph Construction (SCALE=8)
- □ Global STREAM Performance
- □ HPCC RA Performance
- □ HPCC STREAM-EP Performance
- □ HPCC PTRANS Performance
- □ HPCC FFT Performance
- □ HPCC HPL Performance
- □ SSCA#2 (SCALE=8)
- □ LULESH (release)
- □ EP STREAM (fragmented)
- □ Global STREAM
- □ Global STREAM using Promotion
- □ Global STREAM Performance (GB/s)
- □ HPCC RA Performance (GUPS)
- □ HPCC STREAM-EP Performance (GB/s)
- □ HPCC PTRANS Performance (GB/sec)
- □ HPCC FFT Performance (Gflop/s)
- □ HPCC HPL Performance (Gflop/s)
- □ EP STREAM (fragmented) (GB/s)
- □ Global STREAM (GB/s)
- □ Global STREAM using Promotion (GB/s)
- □ SSCA#2 Kernel 4 TEPS (SCALE=8)
- □ LULESH (orig)
- □ LULESH (bradc study version)
- □ miniMD LJ (--size=10) Time
- □ Fannkuch-Redux (n=12)
- □ Binary Trees Shootout Benchmark (n=20)
- ☑ Spectral Norm Shootout Benchmark
- □ Chameneos Redux Shootout Benchmark (n=6,000,000)
- ☑ N-body variations
- ☑ Fasta Shootout Benchmark (n=25,000,000)
- □ Thread Ring Shootout Benchmark (n=50,000,000)
- ☑ Mandelbrot variations
- □ Pi digits variations

http://chapel.sourceforge.net/perf/

Chapel Language Shootout* Entry Underway!



Ubuntu : Intel® Q6600® one core

Computer Language Benchmarks Game [[Play]]

quad-core and one core, x86 and x64 measurements

n-body benchmark N=50,000,000

Each chart bar shows *how many times slower*, one ↓ **n-body** program was, compared to the fastest p

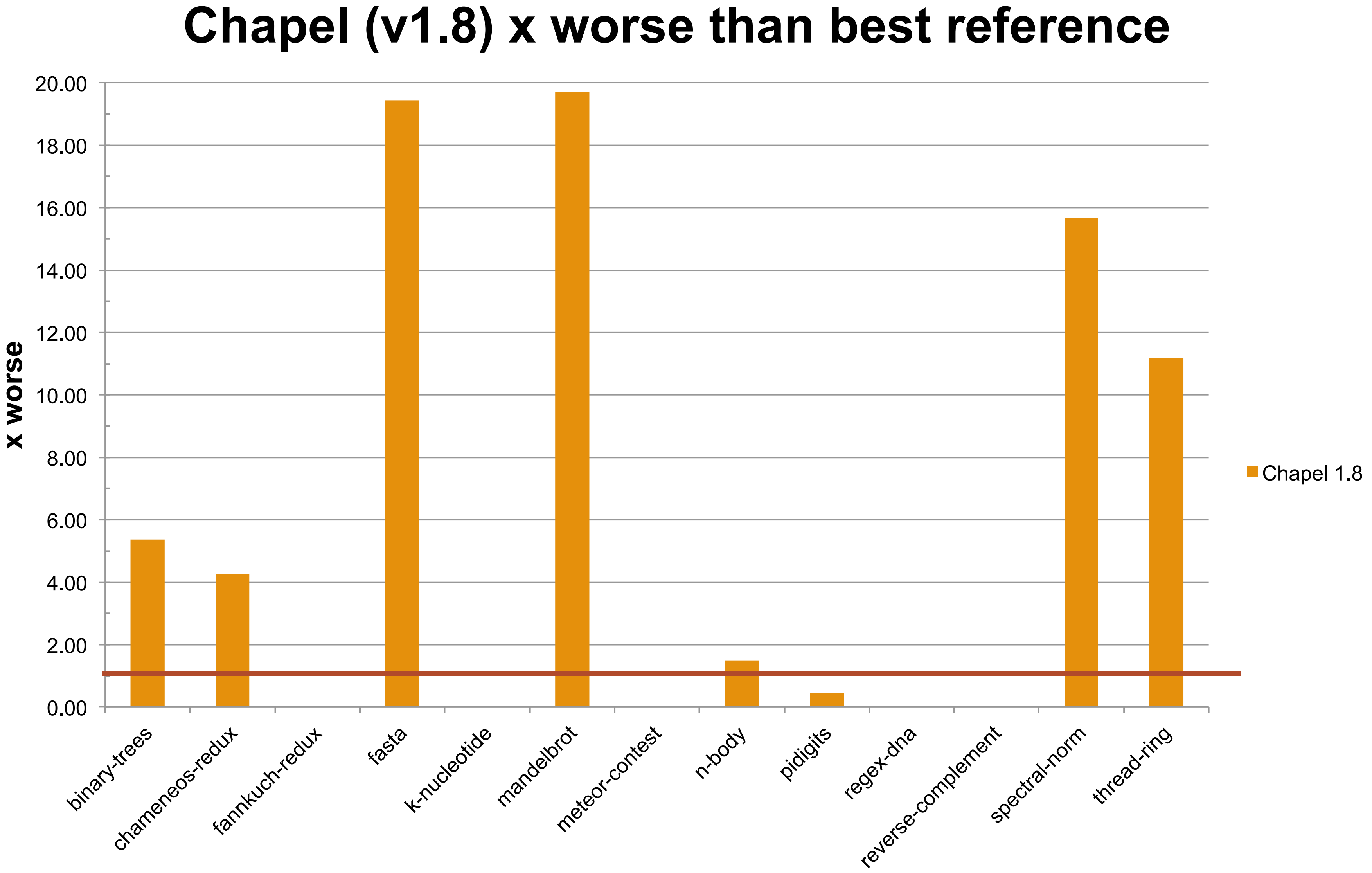
These are not the only programs that could be written. These are not the only compilers and interpret

Column × shows *how many times more* each program used compared to the benchmark program tha

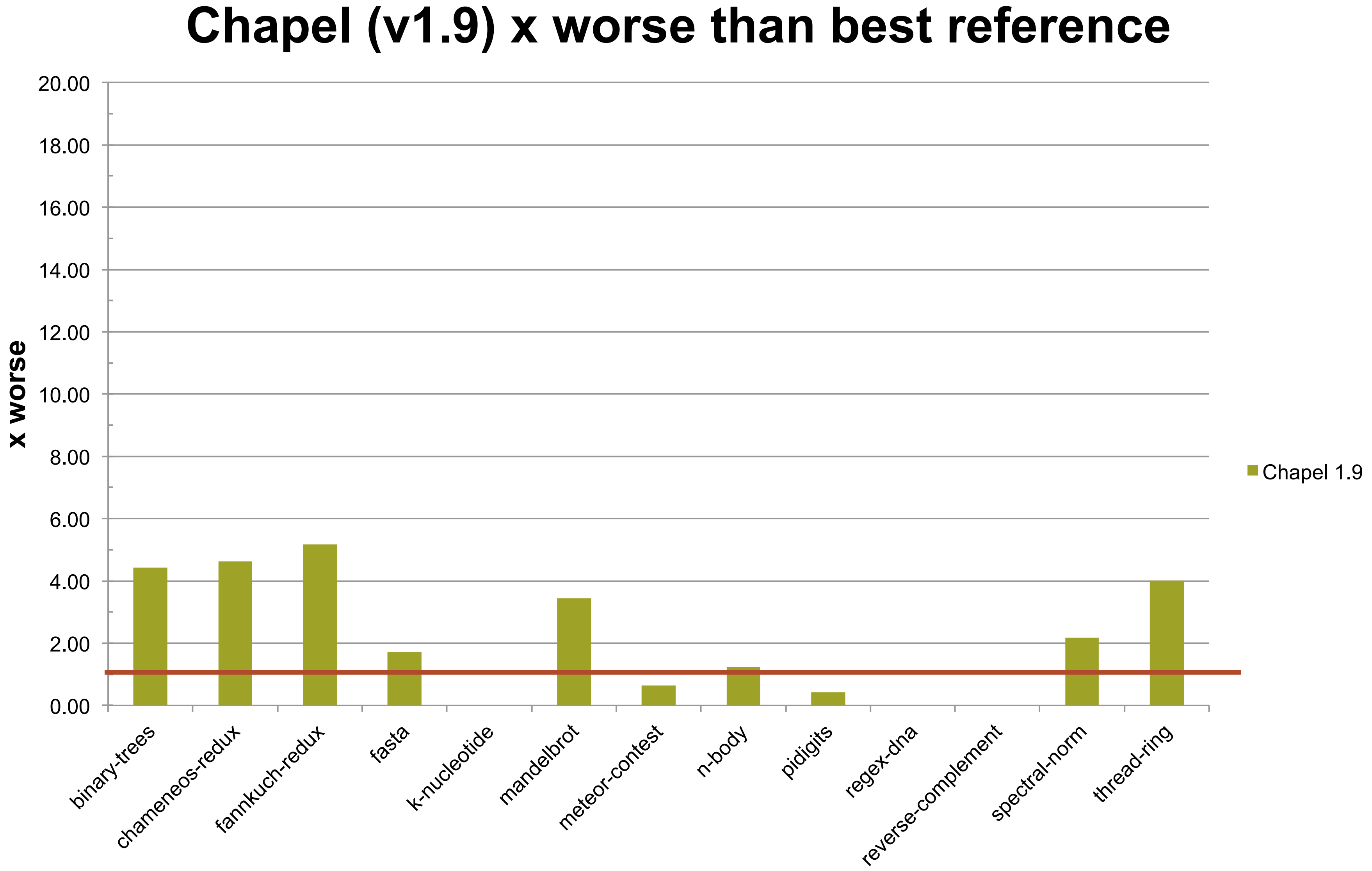
		sort		sort	sort	
×	Program Source Code	CPU secs	Elapsed secs	Memory KB	Code B	≈ CPU Load
1.0	**Fortran** Intel #5	8.57	**8.57**	260	1659	0% 0% 0% 100%
1.1	**C++** g++ #8	9.08	**9.08**	336	1544	0% 0% 1% 100%
1.1	**C++** g++ #7	9.10	9.10	336	1545	2% 0% 5% 100%
1.2	**C** gcc #4	9.91	**9.92**	336	1490	0% 1% 1% 100%
1.2	**C++** g++ #3	9.94	9.95	620	1763	1% 0% 1% 100%
1.5	**C++** g++ #5	12.74	12.75	868	1749	0% 1% 1% 100%

* = ahem… The Computer Language Benchmarks Game
http://benchmarksgame.alioth.debian.org/

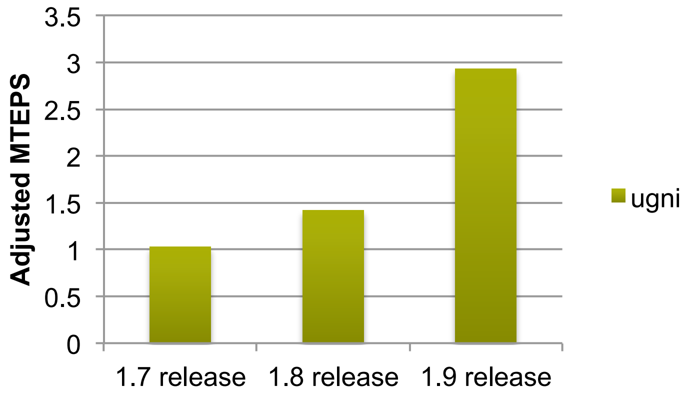
Shootout Performance Summary (v1.8)



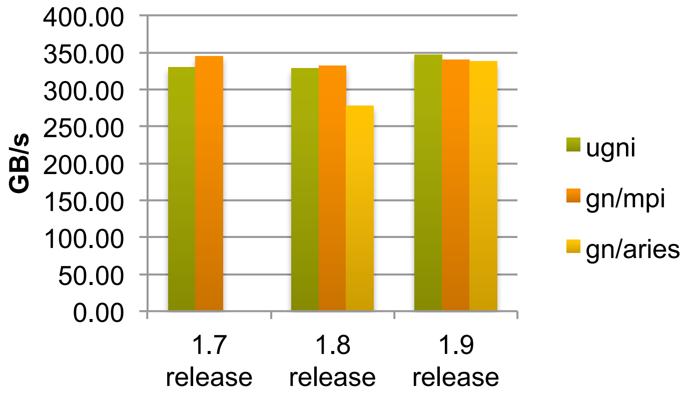
Shootout Performance Summary (v1.9)



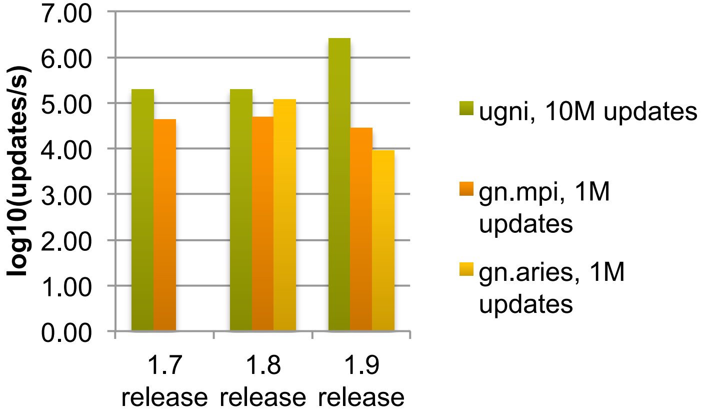
Multilocale Performance is also Improving!



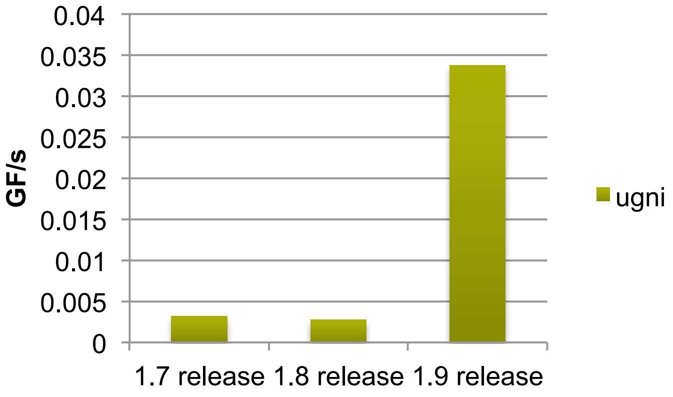
SSCA#2



STREAM Triad - Global



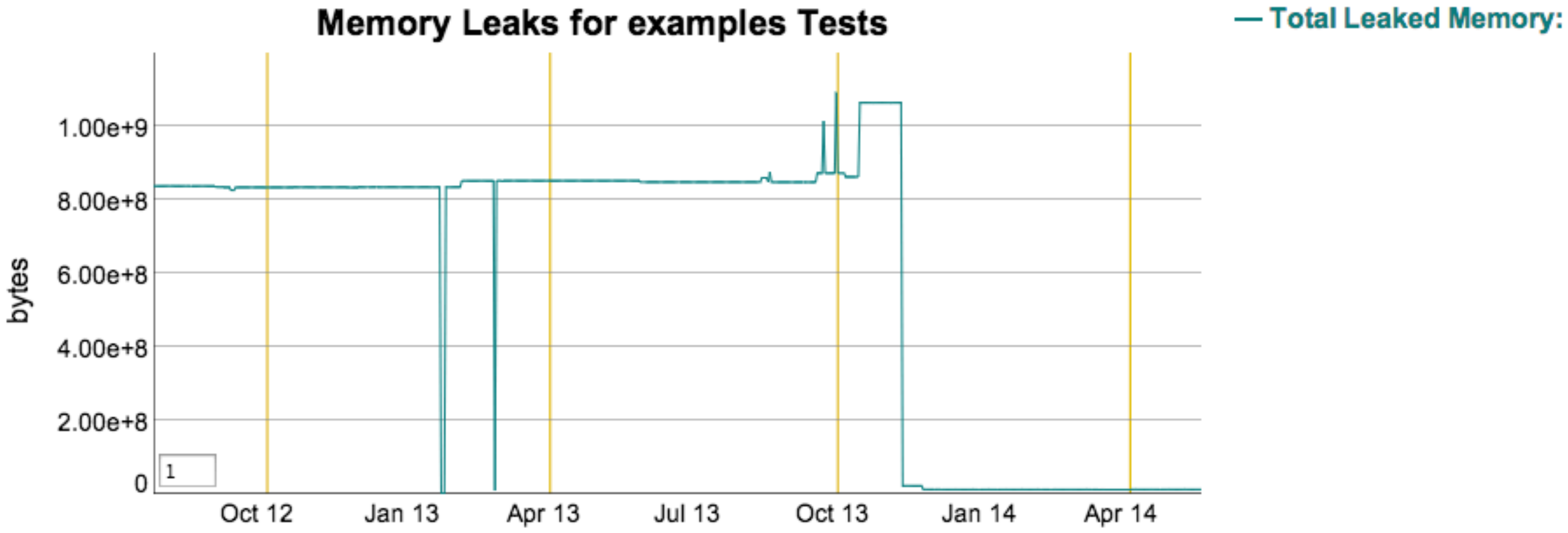
HPCC RA using `on`



HPCC FFT

(Performance graphs – higher is better)

Memory Leaks are Being Plugged!



Copyright 2014 Cray Inc.

Correctness Testing Now Public Too!

- **Nightly regression tests sent to SourceForge mailing lists:**
  - **chapel-test-results-regressions@…** the interesting results
  - **chapel-test-results-all@...** the complete results

- **Also, working on transitioning testing to Jenkins**
  - more holistic, integrated management of testing
  - better historical record of failure cases
  - http://jenkins-ci.org/

http://sourceforge.net/p/chapel/mailman/

Chapel Developers Join 21st Century?!

- **Moving source base from SVN/SourceForge to git/GitHub**

- **Also, working on setting up…**
  - issue tracking
  - task management
  - patch review framework
  - etc.

- **This effort is being led by Thomas Van Doren (here today)**

Change to License/Contributor Agreement!

Historically:

- **License:** BSD
- **Contributor Agreement:** Cray-specific

Plan for version 1.10:

- **License:** Apache
- **Contributor Agreement:** Apache

Rationale:

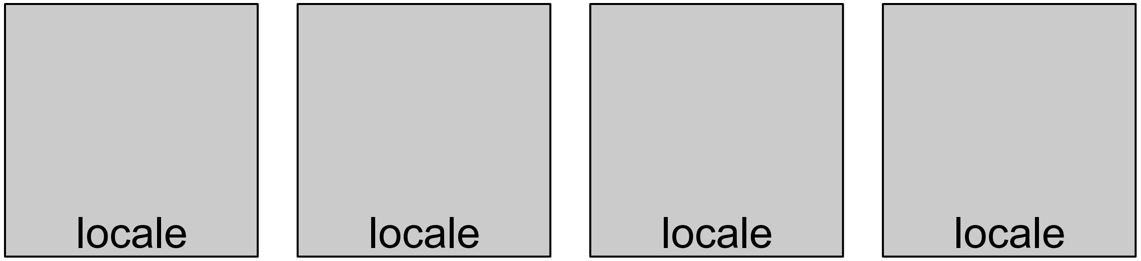
- BSD doesn't have a contributor agreement
- Cray agreement has been a stumbling block for some developers

http://www.apache.org/licenses/LICENSE-2.0.html

User-Defined Hierarchical Locale Models Now Available!

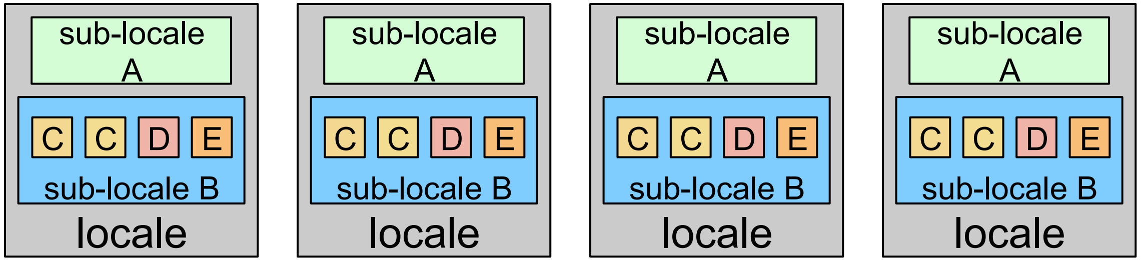
Traditionally:

- Chapel's locales were flat – no locality control within locales
- Tasking and memory interfaces baked into the Chapel compiler



As of v1.8:

- Locales may now contain sublocales
- Users may write their own locale models using Chapel code
- Tasking and memory interfaces defined via such Chapel modules

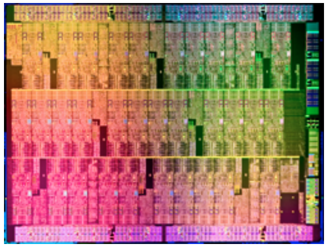


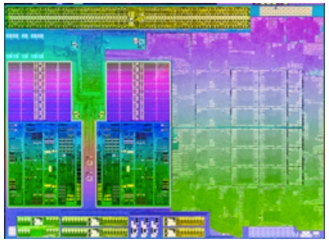
http://chapel.cray.com/presentations/KIISE-KOCSEA_HPC_Workshop2013.pdf

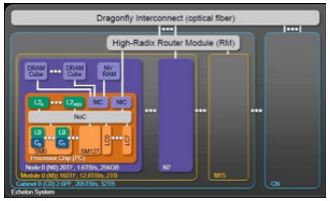
http://svn.code.sf.net/p/chapel/code/trunk/doc/release/technotes/README.localeModels

Chapel: An Exascale Programmer's Dream?

- **Consider an exascale programmer's wishlist:**
  - general types of parallelism
    - task parallelism to offload computations
    - data parallelism for SIMD execution
    - nested parallelism for composition
  - locality control
    - and distinct from parallelism
  - separation of concerns
    - algorithms from implementation (via domain maps, iterators)
  - portability to diverse / unknown hardware architectures
    - user-defined locale models
  - programmability features, to keep sane






Dragonfly Interconnect (optical fiber)
High-Radix Router Module (RM)
DRAM Cube
DRAM Cube
NV RAM
L2
L2
MC
NIC
NoC
L0
L0
C
C
SM0
SM127
LC0
LC7
Processor Chip (PC)
Node 0 (N0) 20TF, 1.6TB/s, 256GB
N7
Module 0 (M0) 160TF, 12.8TB/s, 2TB
M15
Cabinet 0 (C0) 2.6PF, 205TB/s, 32TB
CN
Echelon System

- **Hybrid models can do some of this…**
  - But who wants to use a hybrid model?

- **Main question:** Can Chapel get far enough fast enough?

http://chapel.cray.com/presentations/ChapelForPADAL-distributeme.pdf

http://chapel.cray.com/presentations/ChapelForPGASX-presented.pdf

Chapel: It's not just for HPC anymore!

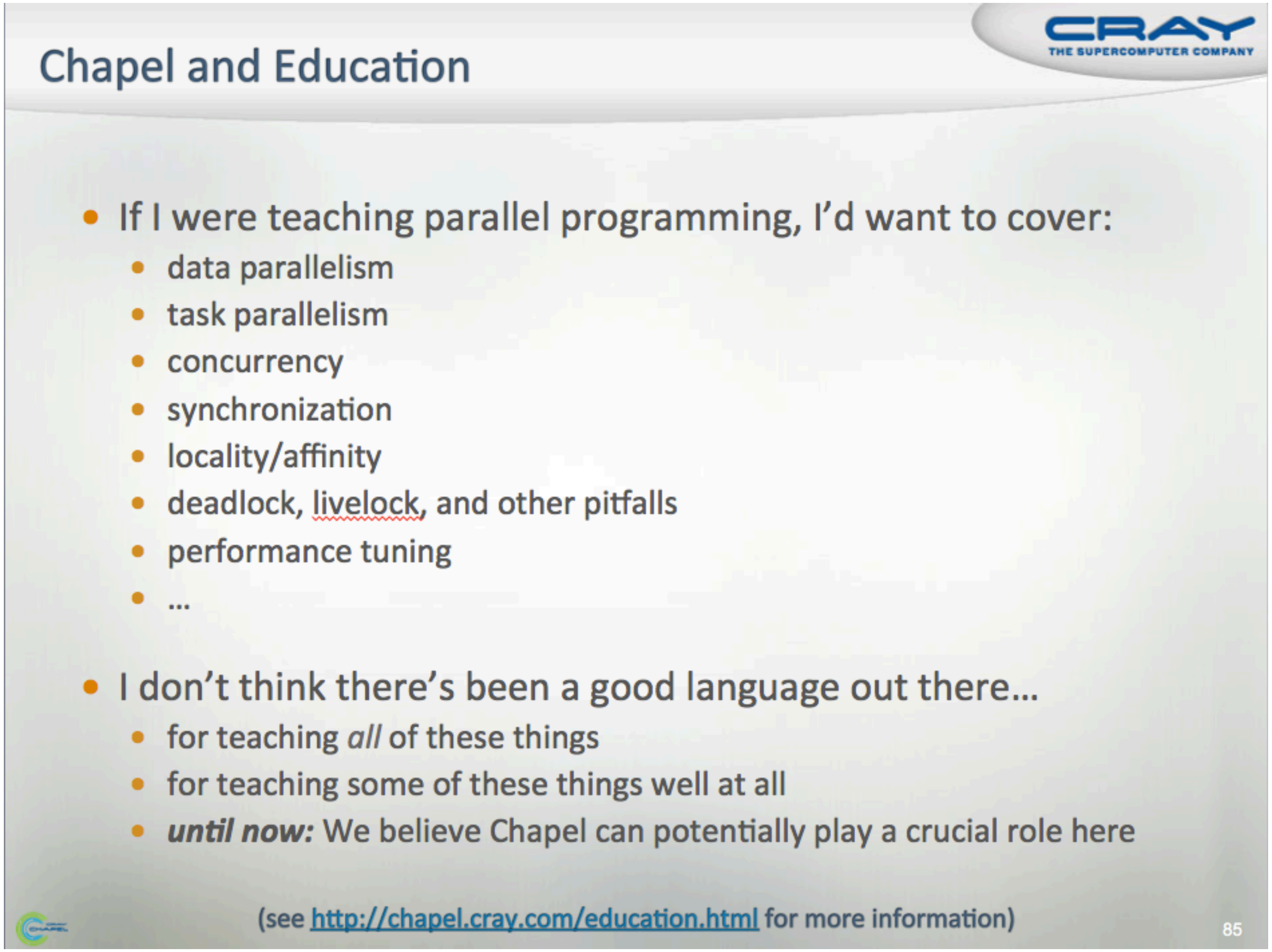
- **"Big data" programmers want productive languages too**
  - MapReduce, Pig, Hive, HBase have their place, but drawbacks too
  - Wouldn't a general, locality-aware parallel language be nice here too?
- **Chapel support for HDFS*: A first step**
  - Developed by Tim Zakian (Indiana University) last summer
  - Returning this summer to take the next steps
- **Questions:**
  - What killer apps/demos to focus on?
  - What other non-HPC communities should we target?

*HDFS = Hadoop Distributed File System

http://chapel.cray.com/presentations/SC13/06-hdfs-ferguson.pdf

Chapel: Attractive for Education!

- For some time now, we've made the following claim:



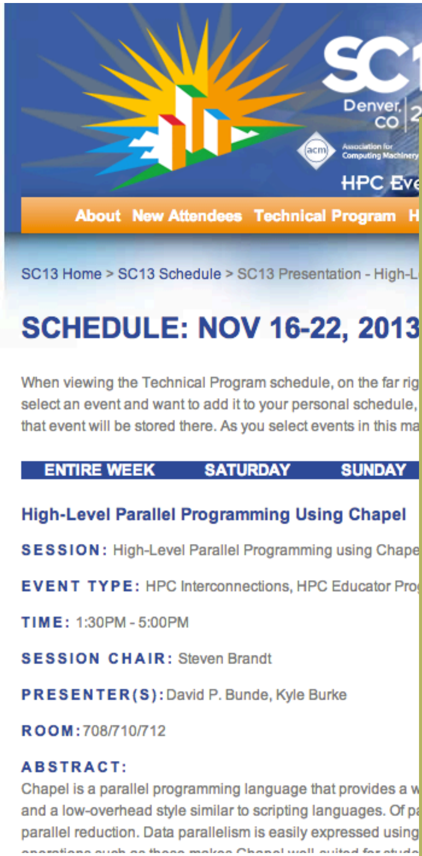
Chapel and Education

- If I were teaching parallel programming, I'd want to cover:
  - data parallelism
  - task parallelism
  - concurrency
  - synchronization
  - locality/affinity
  - deadlock, livelock, and other pitfalls
  - performance tuning
  - …
- I don't think there's been a good language out there…
  - for teaching *all* of these things
  - for teaching some of these things well at all
  - ***until now:*** We believe Chapel can potentially play a crucial role here

(see http://chapel.cray.com/education.html for more information)

Chapel: Attractive for Education!

- And now, educators are making the argument for us:



SC13
Denver, CO 2013
acm Association for Computing Machinery
HPC Eve
About New Attendees Technical Program H

Conference Dates: Nov. 17-22, 2013
Exhibition Dates: Nov. 18-21, 2013

The International Conference for High Performance Computing, Networking, Storage and Analysis

SC13 Home > SC13 Schedule > SC13 Presentation - High-L

SCHEDULE: NOV 16-22, 2013

When viewing the Technical Program schedule, on the far rig select an event and want to add it to your personal schedule, that event will be stored there. As you select events in this ma

ENTIRE WEEK | SATURDAY | SUNDAY

High-Level Parallel Programming Using Chapel

SESSION: High-Level Parallel Programming using Chape

EVENT TYPE: HPC Interconnections, HPC Educator Pro

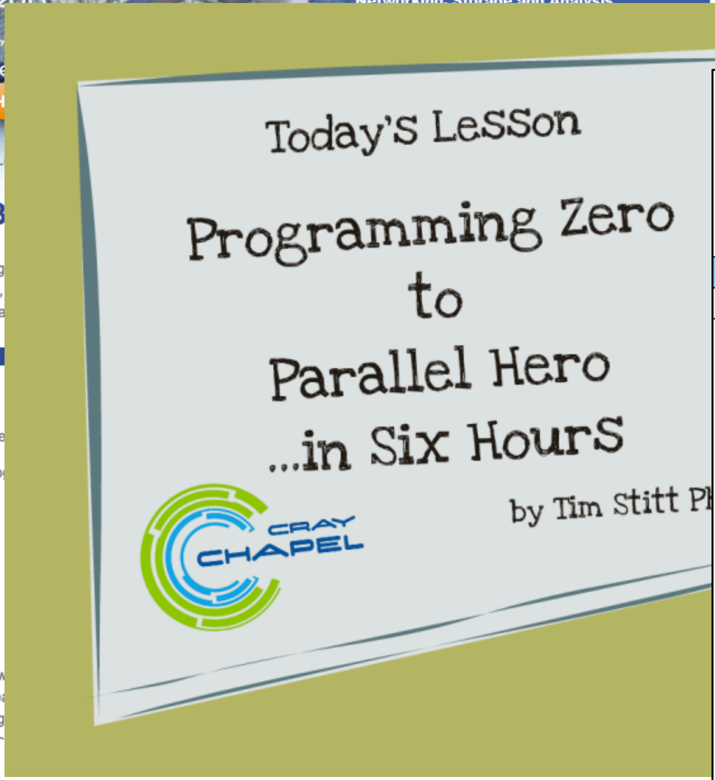
TIME: 1:30PM - 5:00PM

SESSION CHAIR: Steven Brandt

PRESENTER(S): David P. Bunde, Kyle Burke

ROOM: 708/710/712

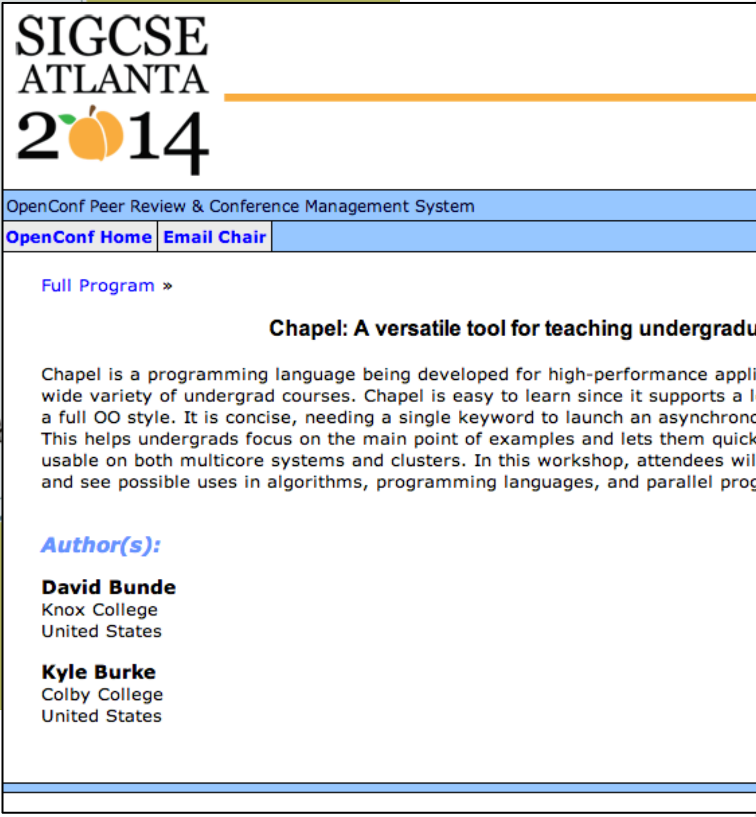
ABSTRACT:
Chapel is a parallel programming language that provides a w and a low-overhead style similar to scripting languages. Of pa parallel reduction. Data parallelism is easily expressed using



Today's Lesson

Programming Zero to Parallel Hero ...in Six Hours

by Tim Stitt Ph



OpenConf Peer Review & Conference Management System

Full Program »

Chapel: A versatile tool for teaching undergradu

Chapel is a programming language being developed for high-performance appli wide variety of undergrad courses. Chapel is easy to learn since it supports a l a full OO style. It is concise, needing a single keyword to launch an asynchrono This helps undergrads focus on the main point of examples and lets them quick usable on both multicore systems and clusters. In this workshop, attendees wil and see possible uses in algorithms, programming languages, and parallel prog

Author(s):

David Bunde
Knox College
United States

Kyle Burke
Colby College
United States

http://chapel.cray.com/education.html

Interactive Chapel?!

- What if you could work with Chapel interactively:

```
chpl> var A: [1..n] real;
OK.
chpl> [i in 1..n] A = i / 2.0;
OK.
chpl> writeln(A);
0.5 1.0 1.5 2.0 2.5 3.0
chpl> proc foo(x) { x *= 2; }
OK.
```

- What if this worked not only on your desktop, but by offloading onto compute nodes as well:

```
chpl> var myLocales = getNodes(100);
OK.
chpl> var MyDist = new Block({1..1000000}, myLocales);
OK.
```

- We've just started a study on such a capability

Working toward "The Chapel Foundation"!

- **If Chapel remains Cray-steered, its chances of succeeding are much lower**

- **The intention has always been to "turn it over to the community" when it's ready**
  - finding the correct timing is tricky

- **We've started the brainstorming process of what such a model would look like ("The Chapel Foundation")**
  - membership roles
  - governance
  - funding models

- **If you have thoughts on this, we're interested in them**

What's Next?

At Cray: Current Tasks

- **Portability:**
  - Support for Intel Phi
- **Performance:**
  - Performance analysis/optimization for NUMA architectures
  - Improved generation of C for loops (including parallelization pragmas)
  - Make Qthreads the default tasking layer
  - Make tcmalloc the default memory allocator
  - Enable processor-specific C back-end optimizations
- **Language/Library Features:**
  - Improve strings
  - Improved memory management capabilities
  - Bit Operations library
  - Improve semantics for variables accessed within ‘forall’
  - Re-implement variable initialization and implement ‘noinit’
- **Code Studies:**
  - Wrap up and submit shootout entry
  - Improved support for stencil computations

Outside of Cray?

- What are you interested in working on?

and/or

- Stay tuned for the rest of the CHIUW talks…

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.

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