Chapel Users Group (CHUG) BoF

Brad Chamberlain, Sung-Eun Choi, Greg Titus
Chapel Team, Cray Inc.
CHUG BoF, SC14
November 19th, 2014
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; domestically & internationally

● A work-in-progress

● Goal: Improve productivity of parallel programming
What does “Productivity” mean to you?

Recent Graduates:
“something similar to what I used in school: Python, Matlab, Java, …”

Seasoned HPC Programmers:
“that sugary stuff that I don’t need because I was born to suffer”
want full control
“to ensure performance”

Computational Scientists:
“something that lets me express my parallel computations
without having to wrestle with architecture-specific details”

Chapel Team:
“something that lets computational scientists express what they want,
without taking away the control that HPC programmers need,
implemented in a language as attractive as recent graduates want.”
Chapel's Implementation

- Being developed as open source at GitHub
  - Licensed as Apache v2.0 software

- Portable design and implementation, targeting:
  - multicore desktops and laptops
  - commodity clusters and the cloud
  - HPC systems from Cray and other vendors
  - in-progress: manycore processors, CPU+accelerator hybrids, …
Outline For CHUG 2014

✓ Welcome / Context

➢ Chapel’s Cadence and Community
  ● Chapel in a Nutshell
  ● Chapel Current Events
  ● Open Q&A
  ● Community Discussion
A Year in the Life of Chapel

● **Two major releases per year** (April / October)
  ● latest release: version 1.10, October 2\textsuperscript{nd}, 2014
  ● ~a month later: detailed release notes
    ● version 1.10 release notes: [http://chapel.cray.com/download.html#releaseNotes](http://chapel.cray.com/download.html#releaseNotes)

● **SC** (Nov)
  ● annual “Lightning Talks” BoF featuring talks from the community
  ● CHUG happy hour (now in its 5\textsuperscript{th} year)
    ● and now a CHUG BoF!
  ● plus tutorials, panels, BoFs, posters, educator sessions, exhibits, …

● **CHIUW:** Chapel Implementers and Users Workshop (May-June)
  ● talk-based workshop focusing on community efforts

● **Talks, tutorials, research visits, blogs, …** (year-round)
Chapel at SC14

Chapel Tutorial (Sun @ 8:30)
“A Computation-Driven Introduction to Parallel Computing in Chapel”

Hierarchical Locales Exhibit at Emerging Technologies Booth (all week, booth #233)
poster staffed by members of the Chapel team

4th Annual Chapel Lightning Talks BoF (Tues @ 12:15, room 293)
5-minute talks on Chapel + HSA, HDFS/Lustre/cURL, tilings, LLVM, ExMatEx, Python

Talk on Hierarchical Locales (Tues @ 4:30, Emerging Technologies Theater, booth #233)
“Chapel Hierarchical Locales: Adaptable Portability for Exascale Node Architectures”, Greg Titus (Cray)

Poster on Advanced Tilings in Chapel (Tues @ 5:15, New Orleans Theater Lobby)
“Orthogonal Scheduling of Stencil Computations with Chapel Iterators”, Ian Bertolacci (Colorado State)

Chapel Users Group (CHUG) BoF (Wed @ 5:30, room 383-84-85)
Chapel overview and current events, followed by community Q&A and discussion

5th Annual CHUG Happy Hour (Wed @ 7:15, Mulate’s at 201 Julia St)
social gathering just across the way; open to general public, dutch treat

Participation in other BoFs:
● LLVM in HPC (Tues @ 12:15, room 283-84-85)
● Programming Abstractions for Data Locality (Wed @ 12:15, room 391-92)
● PGAS: Partitioned Address Space Programming Model (Wed @ 12:15, room 273)
Chapel Lightning Talks 2014 Lineup

Chapel Overview
   Greg Titus, Cray Inc.

CoMD in Chapel: The Good, the Bad, and the Ugly
   David Richards, Lawrence Livermore National Laboratory

Chapel for Python Programmers
   Simon Lund, University of Copenhagen

Chapel Iterators: Providing Tiling for the Rest of Us
   Ian Bertolacci, Colorado State University

Chapel I/O: Getting to Your Data Wherever It Is
   Tim Zakian, Indiana University

LLVM-based Communication Optimizations for Chapel
   Akihiro Hayashi, Rice University

COHX: Chapel on HSX + XTQ
   (Adventures of a PGAS Language in a Heterogenous World)
   Deepak Majeti, Rice University
CHIUW 2014 Talks and Speakers

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

Keynote: Walking to the Chapel
Robert Harrison, Stony Brook University / Brookhaven National Laboratory

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
Submitted as a workshop to PLDI
- held with FCRC in Portland, June 13-17, 2015

Speakers:
- <your name here?>
- extended abstract submissions due March 2015
Just Who Are These Chapel Users Anyway?

Truthfully, most Chapel users are potential/hopeful users

Most are waiting, typically for…
   …improved performance
   …improved stability
   …reassurance that they won’t be the only users

That said, some early user categories include:
   ● educators and students
   ● computer science researchers
   ● Python users looking for something compiled, scalable, productive
   ● HPC programmers evaluating new programming models
Roll Call

Who here would self-classify as…

…Chapel users?
…potential future Chapel users?
…simply curious about Chapel?
…Chapel developers?
…those who were simply intrigued by the CHUG happy hour afterwards?
Outline For CHUG 2014

✓ Welcome / Context
✓ Chapel’s Cadence and Community

➢ Chapel in a Nutshell
  ● Chapel Current Events
  ● Open Q&A
  ● Community Discussion
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 a new system architecture, network, software stack…
  - 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**: Support multiple tiers of features

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

*Chapel language concepts*

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine

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

Chapel language concepts

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control

Lower-level Chapel

Target Machine
Chapel in a Nutshell: Base Language

Static Type Inference for:
- arguments
- return types
- variables

CLU-style iterators

```
iter fib(n) {
    var current = 0,
        next = 1;
    for i in 1..n {
        yield current;
        current += next;
        current <=> next;
    }
}
```

swap operator

```
for (i, f) in zip(0..#n, fib(n)) do
    writeln("fib #", i, " is ", f);
```

zippered iteration

range types and operators

```
fib #0 is 0
fib #1 is 1
fib #2 is 1
fib #3 is 2
fib #4 is 3
fib #5 is 5
fib #6 is 8
...```
Chapel in a Nutshell: Task Parallelism, Locality

High-Level Task Parallelism

Abstraction of System Resources

Locality/Affinity Control

```
taskParallel.chpl

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

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

Chapel language concepts

- Domain Maps
- Data Parallelism
- Task Parallelism
- Base Language
- Locality Control
- Target Machine

Higher-level Chapel

Copyright 2014 Cray Inc.
Chapel in a Nutshell: Data Parallelism

**dataParallel.chpl**

```chapel
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
```
Parallelism and Locality: Orthogonal in Chapel

- This is a **parallel**, but local program:

```chapel
begin writeln("Hello world!");
writeln("Goodbye!");
```

- This is a **distributed**, but serial program:

```chapel
writeln("Hello from locale 0!");
on Locales[1] do writeln("Hello from locale 1!");
writeln("Goodbye from locale 0!");
```

- This is a **distributed, parallel** program:

```chapel
begin on Locales[1] do writeln("Hello from locale 0!");
on Locales[2] do begin writeln("Hello from locale 1!");
writeln("Goodbye from locale 0!");
```
For More Information: Suggested Reading

**Overview Papers:**

  - *a detailed overview of Chapel’s history, motivating themes, features*

  - *a higher-level overview of the project, summarizing the HPCS period*
Blog Articles:

  - a short-and-sweet introduction to Chapel

- **Why Chapel?** (part 1, part 2, part 3), Cray Blog, June-October 2014.
  - a recent series of articles answering common questions about why we are pursuing Chapel in spite of the inherent challenges

  - a series of technical opinion pieces designed to combat standard arguments against the development of high-level parallel languages
Outline For CHUG 2014

✓ Welcome / Context
✓ Chapel’s Cadence and Community
✓ Chapel in a Nutshell
➢ Chapel Current Events
● Open Q&A
● Community Discussion
Chapel Current Events

(meet “Chappie”!)
NEW ORLEANS SHINES FOR STAR-STUDDED WEDDING

TRUE LOVE

SAINTS FAN DEFENDS HIS METHOD OF HAVING A BALL

‘THE MARDI GRAS INSTINCT KICKED IN’

Saints fan Tony Williams reacts to negative responses from other fans at the Superdome on Sunday after he intercepted a souvenir intended for Bengals fan Christina Barrett.

By Amos Morale III
Staff writer

When Bengals tight end Jermaine Gresham tossed a ball into the stands after a touchdown against the Saints on the Internet almost instantaneously, Williams and the Bengals’ fan, Christina Barrett, were all over social media.

Williams said Monday he was a bit surprised and “a little hurt” by the reaction, which mostly centered on him as a will.
HPCS may be over, But Chapel’s Alive and Well
HPCS is over, but Chapel is Alive and Well

- Based on positive user response to Chapel under HPCS, Cray is undertaking a five-year effort to improve it
  - we’re currently partway into our second year

- Focus Areas:
  1. Improving **performance** and scaling
  2. Fixing immature aspects of the language and implementation
     - e.g., strings, RAII/memory model, error handling, …
  3. Porting to emerging architectures
     - Intel Phi, accelerators, heterogeneous processors and memories, …
  4. Improving **interoperability**
  5. Growing the Chapel user and developer **community**
     - including non-scientific computing communities
  6. Transitioning the **governance** to neutral, external group
Chapel Team at Cray Has Doubled in Size
We’ve doubled the size of the Chapel Team at
Chapel Team at Cray Hiring SW Dev
Broader Chapel Community Has Also Grown
The Broader Chapel Community Has Also Grown

(there’s been an uptick in interest from industrial users/developers as well)

http://chapel.cray.com/collaborations.html
Chapel version 1.10 Now Available
Chapel version 1.10 is now available

● **Highlights Include:**
  ● lighter-weight tasking via Sandia’s Qthreads
  ● initial support for Intel Xeon Phi Knights Corner (KNC)
  ● renewed focus on standard libraries
  ● support for Lustre and cURL-based data channels
  ● expanded array capabilities
  ● improved semantic checks, bug fixes, third-party packages, …
  ● significant performance improvements…

https://github.com/chapel-lang/chapel/releases/tag/1.10.0
Chapel Shared-Mem Execution Time Improving
Execution Time is Improving (lower is better)
Chapel Performance Tracking Data Now Publicly Revealed
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

http://chapel.sourceforge.net/perf/
Chapel Team Planning to Enter Shootout
Chapel Language Shootout* Entry Underway

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

Copyright 2014 Cray Inc.
Shootout benchmark performance (lower is better)

X worse than reference

Chapel 1.8 release
Chapel 1.9 release
Chapel 1.10 release
Shootout: Performance for 1.9 vs. 1.10 (gcc 4.7)

X worse than reference

Chapel 1.9 release
Chapel 1.10 release
Distributed Memory Perf/Scalability Also Improving
HPCC multilocal performance (higher is better)

STREAM (GB/s)

RA, atomic version (GUPS)

HPL, study version (GF/s)

FFT (GF/s)
STREAM Scalability (1.9 vs. 1.10, higher=better)
RA Scalability (1.9 vs. 1.10, higher is better)
Chapel’s Memory Leaks Being Plugged
Memory Leaks over time (lower is better)

Memory Leaks for examples Tests

- Total Leaked Memory:

http://chapel.sourceforge.net/perf/
Correctness Testing Results Made Public
Correctness Testing Now Public Too

Nightly regression test results are sent to public SourceForge mailing lists:

- chapel-test-results-regressions@... the interesting results
- chapel-test-results-all@... the complete results

http://sourceforge.net/p/chapel/mailman/
Chapel Developers Join 21st Century
Chapel Developers Join 21st Century

- Migrated from SVN/SourceForge to Git/GitHub
- Converted testing from crontabs to Jenkins
- Began using Travis for pre-commit sanity checks
- Began using Coverity Scan to catch code quality issues
- Started tracking tasks in Pivotal
- Kicked off a Facebook page
- Started a #chapel-developers IRC channel
- Created/Owned a Chapel project in OpenHUB

- Next up: modern, online documentation and more…
Chapel Adopts Apache License, Contributor Agrmt
Chapel is now Apache

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

As of version 1.10:
- **License**: Apache v2.0
- **Contributor Agreement**: Apache v2.0

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
Hierarchical Locale Models Now In-Use
Hierarchical Locale Models Now In Use

Traditionally:
- Chapel’s locales have been 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

See our emerging technologies exhibit here at SC14!
Chapel: An Exascale Programmer’s Dream Come True?
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
    ● distinct from parallelism
  ● separation of algorithms from mapping to system
    ● domain maps, iterators
  ● portability to diverse / unknown hardware architectures
    ● user-defined locale models
  ● programmability features, to keep sane

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

http://chapel.cray.com/presentations/ChapelForPADAL-distributeme.pdf
http://chapel.cray.com/presentations/ChapelForPGASX-presented.pdf
Chapel: Not Just for HPC Anymore?
Chapel: It’s not just for HPC anymore

● “Big data” programmers want productive languages too
  ● MapReduce, Pig, Hive, HBase have their place, but also drawbacks
  ● 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
  ● This summer: extended support to include Lustre, cURL

● Questions:
  ● What killer apps/demos to focus on?

*HDFS = Hadoop Distributed File System

Chapel: Attractive for Education
Chapel: Attractive for Education

For some time, we’ve claimed Chapel is ideal for education:

Chapel and Education

- When teaching parallel programming, I like 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 play a crucial role here

(see http://chapel.cray.com/education.html for more information and http://cs.washington.edu/education/courses/csep524/13wi/ for my use of Chapel in class)
Chapel: Attractive for Education

And now, educators are helping make the argument for us:

http://chapel.cray.com/education.html
Interactive Chapel?!
Interactive Chapel

● What if you could work with Chapel interactively:

```chapel
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:

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

● We’ve recently started an effort to implement such a capability
Laying the Foundation for Chapel Foundation
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
For More Information: Online Resources

Chapel project page: [http://chapel.cray.com](http://chapel.cray.com)
- overview, papers, presentations, language spec, ...

Chapel Facebook page: [http://fb.com/ChapelLanguage](http://fb.com/ChapelLanguage)

Chapel GitHub page: [https://github.com/chapel-lang](https://github.com/chapel-lang)
- download 1.10.0 release, browse source repository

Chapel SourceForge page: [https://sourceforge.net/projects/chapel/](https://sourceforge.net/projects/chapel/)
- join community mailing lists; alternative release download site

Mailing Lists:
- chapel_info@cray.com: contact the team at Cray
- chapel-announce@lists.sourceforge.net: list for announcements only
- 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
Questions? Topics For Discussion?
CHIUW Community Discussion Questions
What are the three most important things that Chapel and the community need to do in the next four years in order to increase the user base and maximize the chances of success? Alternatively, what are the top three things currently inhibiting the rate of Chapel adoption?

contributed by: Pete Ungaro, Peg Williams (Cray Inc.)
Is there a *killer* application space for Chapel (potentially outside of HPC) with strong community support that could help build support for the language? If so, how do we tap into that community?

*contributed by: Rob Neely (LLNL)*
Are there lessons from the adoption (or failure) of other largely community-supported languages (e.g. Python, Ruby) that Chapel could learn from to attract wider support, and avoid becoming a footnote in the history of programming languages? Do you believe this is possible in the HPC field?

contributed by: Rob Neely (LLNL)
Jens Brietbart (Technische Universität München)
What future capabilities of expected, presumed, and rumored large scale computing systems (supercomputers, clusters, cloud, etc.) do you see directly impacting the ability of Chapel to provide acceptable performance?

contributed by: Richard Barrett (Sandia), Thomas Van Doren (Cray Inc.)
What model of collaboration should we adopt as we move Chapel to become a community-controlled project? Should there be separate models for the language specification and the actual implementation?

contributed by: Tom Hildebrandt (Cray Inc.)
Domain Specific Languages (DSLs) have regained popularity recently. Could Chapel and its multi-resolution philosophy be used as the foundation for building DSLs?

contributed by: Peg Williams (Cray Inc.)
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.

Copyright 2014 Cray Inc.