



# State of the Chapel Project 2016

Brad Chamberlain  
Chapel Team, Cray Inc.  
CHIUW 2016, May 27, 2016



---

COMPUTE | STORE | ANALYZE

# 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.



# A Year in the Life of Chapel

- **Two major releases per year** (April / October)
  - ~a month later: detailed release notes available online
- **CHIUW:** Chapel Implementers and Users Workshop (May/June)
- **SC** (Nov)
  - tutorials, BoFs, panels, posters, educator sessions, exhibits, ...
  - annual **CHUG** (Chapel Users Group) happy hour
  - For SC16: Hope to re-establish the **Chapel Lightning Talks** BoF
    - concept: propose 4 (CHIUW?) talks at submission time, 2 wildcards
- **Talks, tutorials, collaborations, social media, ...** (year-round)



# A Year in the Life of Chapel

- **Two major releases per year** (April / October)
  - ~a month later: detailed release notes available online
- **CHIUW:** Chapel Implementers and Users Workshop (May/June)
- **SC** (Nov)
  - tutorials, BoFs, panels, posters, educator sessions, exhibits, ...
  - annual **CHUG** (Chapel Users Group) happy hour
  - For SC16: Hope to re-establish the **Chapel Lightning Talks BoF**
    - concept: propose 4 (CHIUW?) talks at submission time, 2 wildcards
- **Talks, tutorials, collaborations, social media, ...** (year-round)



# Chapel Releases since CHI UW 2015



COMPUTE

|

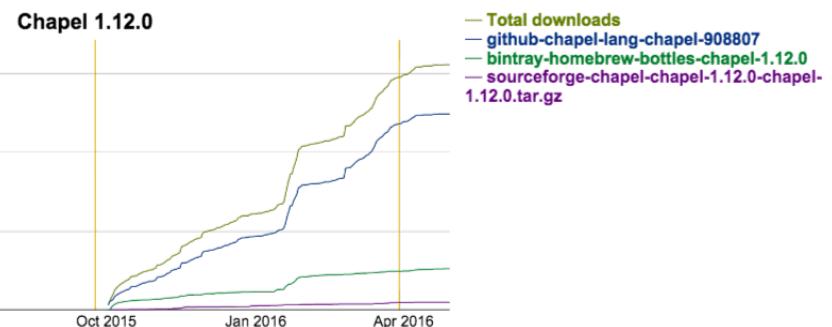
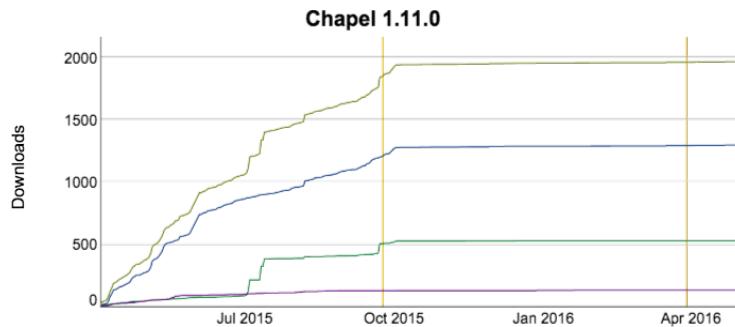
STORE

|

ANALYZE

# Releases since CHIUW 2015

- Since last year, two new versions of Chapel released:
  - 1.12:** October 1, 2015
  - 1.13:** April 7, 2016
- Significant progress in all areas
  - language, library, docs, performance, memory, portability, bugs, ...
- ~3500 downloads in past year for prior two releases:



# Contributors to the Past Year's Releases

## Contributors to 1.12 / 1.13:

- Ben Albrecht, Cray Inc.
- Ian Bertolacci, Cray Inc. / Colorado State University
- Kyle Brady, Cray Inc.
- Paul Cassella, Cray Inc.
- Brad Chamberlain, Cray Inc.
- Sung-Eun Choi, Cray Inc.
- Lydia Duncan, Cray Inc.
- Michael Ferguson, Cray Inc.
- Ben Harshbarger, Cray Inc.
- Tom Hildebrandt, Cray Inc.
- David Iten, Cray Inc.
- Przemysław Leśniak, individual contributor
- Vassily Litvinov, Cray Inc.
- Tom MacDonald, Cray Inc.
- Cory McCartan, Cray Inc.
- Damian McGuckin, Pacific Engineering Systems International
- Phil Nelson, Western Washington University / Cray Inc.
- Michael Noakes, Cray Inc.
- Joshua Olson, individual contributor
- Konstantina Panagiotopoulou, individual contributor
- Nicholas Park, DOD
- Elliot Ronaghan, Cray Inc.
- Kushal Singh, individual contributor
- George Stelle, Sandia National Laboratories
- Chris Taylor, DOD
- Akash Thorat, individual contributor
- Greg Titus, Cray Inc.
- Thomas Van Doren, Cray Inc.
- Tony Wallace, Cray Inc.



# Contributors to the Past Year's Releases

## Contributors to 1.12 / 1.13:

- Ben Albrecht, Cray Inc.
- Ian Bertolacci, Cray Inc. / Colorado State University
- Kyle Brady, Cray Inc.
- Paul Cassella, Cray Inc.
- Brad Chamberlain, Cray Inc.
- Sung-Eun Choi, Cray Inc.
- Lydia Duncan, Cray Inc.
- Michael Ferguson, Cray Inc.
- Ben Harshbarger, Cray Inc.
- Tom Hildebrandt, Cray Inc.
- David Iten, Cray Inc.
- Przemysław Leśniak, individual contributor
- Vassily Litvinov, Cray Inc.
- Tom MacDonald, Cray Inc.
- Cory McCartan, Cray Inc.
- Damian McGuckin, Pacific Engineering Systems International
- Phil Nelson, Western Washington University / Cray Inc.
- Michael Noakes, Cray Inc.
- Joshua Olson, individual contributor
- Konstantina Panagiotopoulou, individual contributor
- Nicholas Park, DOD
- Elliot Ronaghan, Cray Inc.
- Kushal Singh, individual contributor
- George Stelle, Sandia National Laboratories
- Chris Taylor, DOD
- Akash Thorat, individual contributor
- Greg Titus, Cray Inc.
- Thomas Van Doren, Cray Inc.
- Tony Wallace, Cray Inc.

17 Cray employees, 2 Cray interns, 10 external contributors

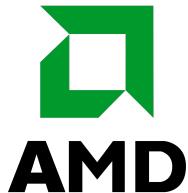


# The Chapel Team at Cray (May 2016)





# Chapel Community R&D Efforts



THE GEORGE  
WASHINGTON  
UNIVERSITY  
WASHINGTON, DC



Lawrence Berkeley  
National Laboratory



(and several others, some of whom you will hear from today...)

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



COMPUTE

STORE

ANALYZE

# Key Language Improvements (1.12 & 1.13)

- **Memory Consistency Model:** specification greatly improved
- **Namespace control:** public/private, filtered module ‘use’s  
(see Lydia Duncan’s talk this afternoon)
- **Memory management of records:** huge improvements
- **Strings:** vastly improved
  - simplified implementation
  - virtually leak-free (a long-term pain-point for users)
  - significant performance improvements
  - standard library of operations



# String Improvements: Leaks / Performance

New string implementation was enabled on December 9<sup>th</sup>

In our test suite...

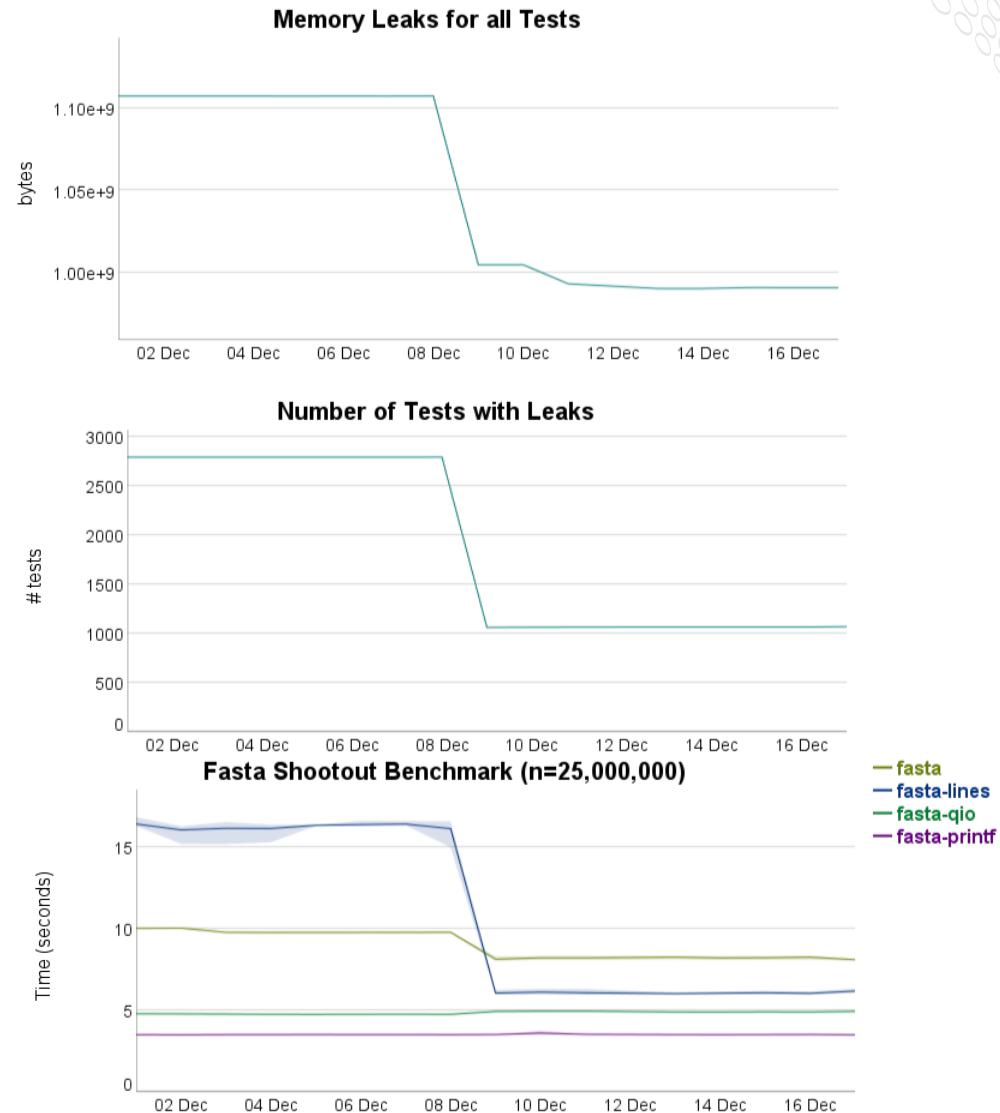
...string-related memory leaks went from 123MB to 22MB

- by 1.13, down to ~300B

...# of tests with leaks reduced by 2.6x

...*fasta-lines* version of CLBG benchmark sped up 2.7x

- another version improved 20%



# String Improvements: Standard Library

## String routines:

this()	// <i>substring</i>	isEmptyString()
these()	// <i>iterate over chars</i>	isUpper()
startsWith()		isLower()
endsWith()		isSpace()
find()		isAlpha()
rfind()		isDigit()
count()		isAlnum()
replace()		isPrintable()
split()		isTitle()
join()		toLowerCase()
strip()		toUpperCase()
partition()		toTitle()
localize()		capitalize()
c_str()		+ , += , * , == , != , <= , ...

# Other Key Library Improvements (1.12 & 1.13)

- **Notable new modules:**

**Spawn**: subprocess control

**Barrier**: barrier synchronization

**Reflection**: meta-programming capabilities

**PCGRandom**: PCG-based random number generation

**LAPACK, LinearAlgebraJAMA**: linear algebra packages

- **Notably improved modules:**

**Random**: supports extended interface and PCG-based algorithms

**FileSystem**: now complete

# Key Documentation Improvements (1.12 & 1.13)

## CHIUW 2015: Online docs had just been launched

- strong positive response from users

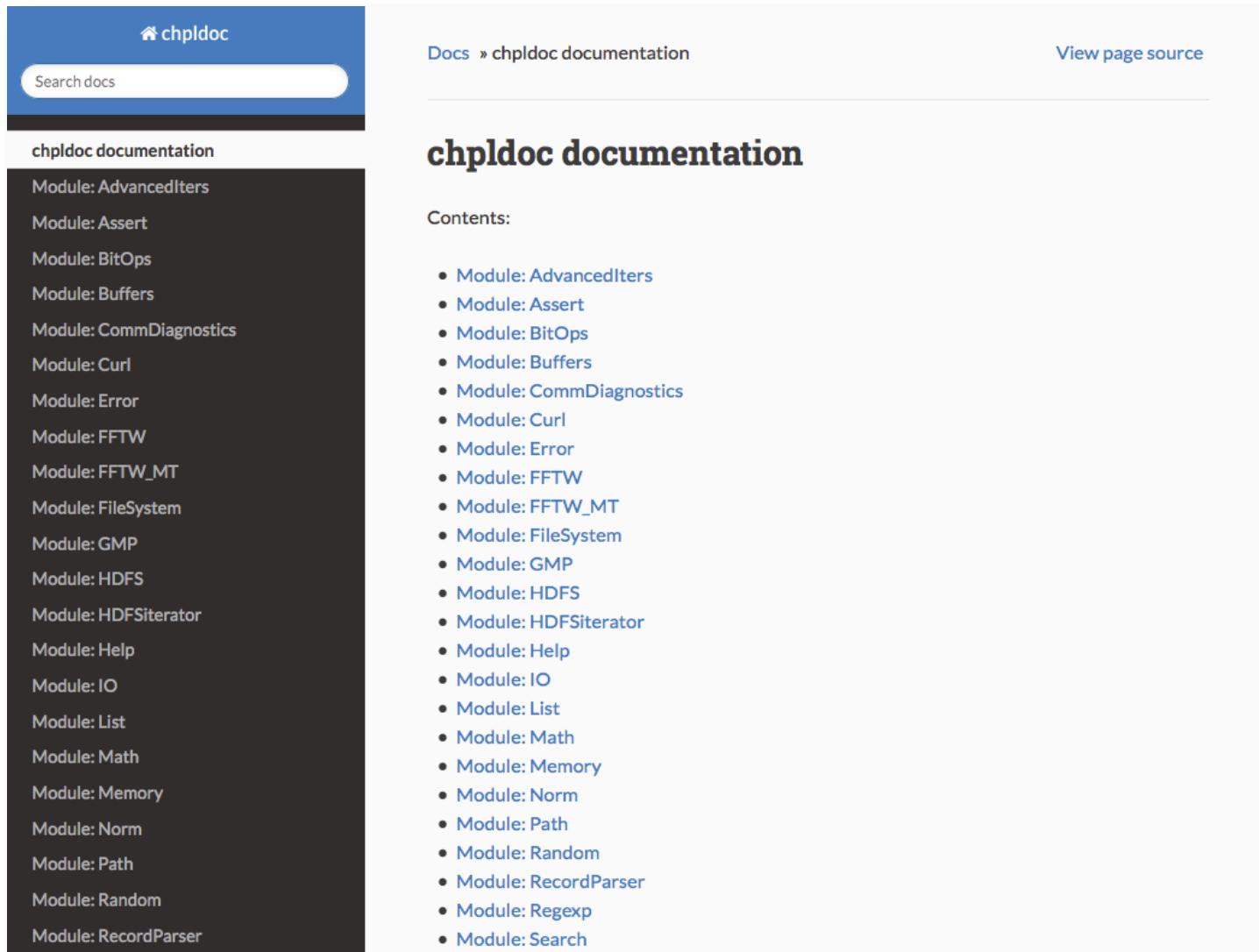
## CHIUW 2016: Docs significantly expanded (125+ pages):

- README-based files from release
- standard distributions / layouts
- “built-in” types / functions
- ‘chpl’ and ‘chpldoc’ man pages
- started writing a Chapel users guide

...also significantly reorganized for clarity

(see following slides or <http://chapel.cray.com/docs/latest/>)

# Online Docs landing page (version 1.11)



The screenshot shows a documentation landing page for the `chpldoc` module. At the top, there's a header bar with the title "chpldoc" and a search bar labeled "Search docs". Below the header is a sidebar titled "chpldoc documentation" containing a list of modules. The main content area shows the "chpldoc documentation" page with a "Contents:" section and a list of module links.

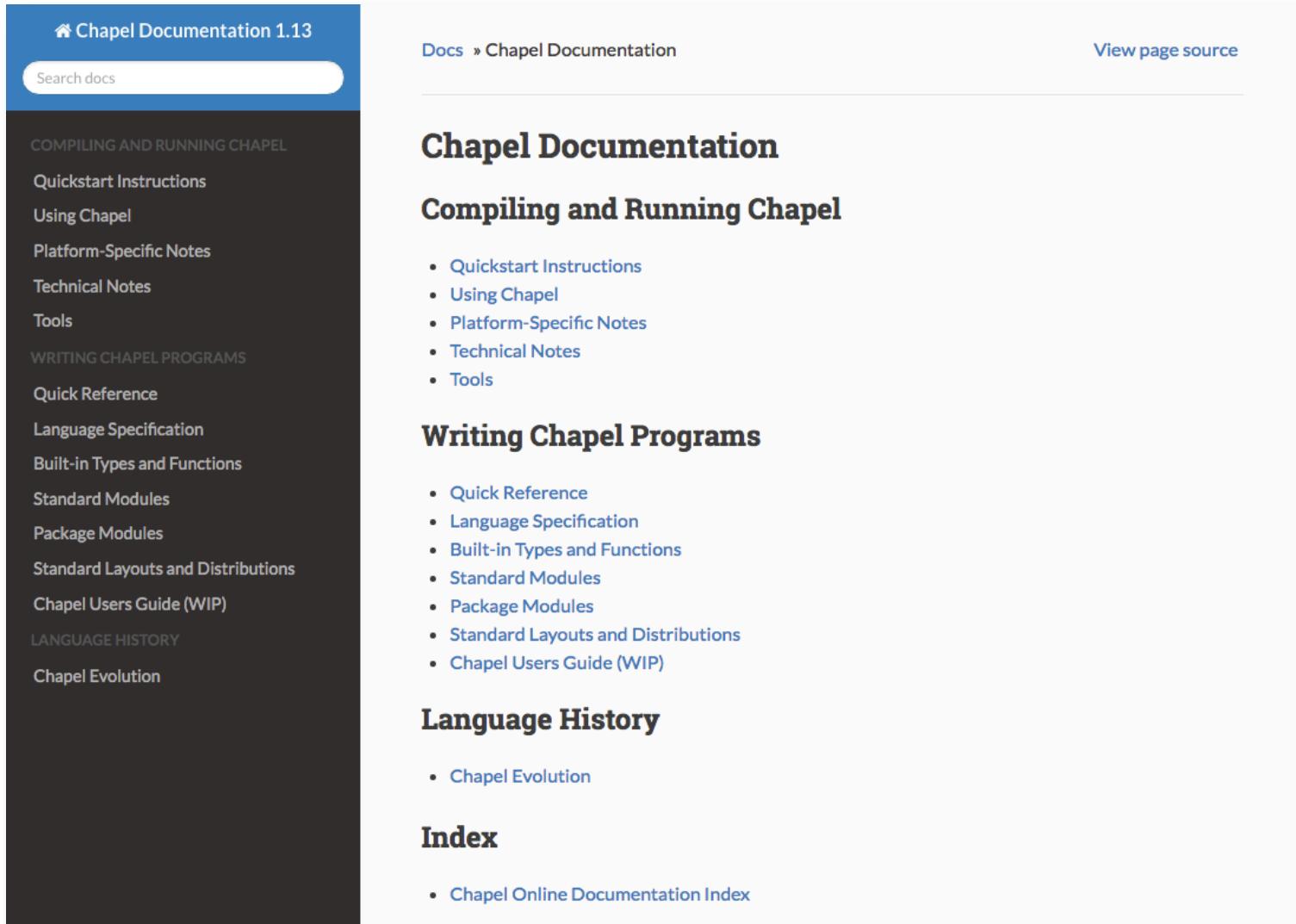
Docs » chpldoc documentation [View page source](#)

## chpldoc documentation

Contents:

- [Module: AdvancedIter](#)
- [Module: Assert](#)
- [Module: BitOps](#)
- [Module: Buffers](#)
- [Module: CommDiagnostics](#)
- [Module: Curl](#)
- [Module: Error](#)
- [Module: FFTW](#)
- [Module: FFTW\\_MT](#)
- [Module: FileSystem](#)
- [Module: GMP](#)
- [Module: HDFS](#)
- [Module: HDFSIterator](#)
- [Module: Help](#)
- [Module: IO](#)
- [Module: List](#)
- [Module: Math](#)
- [Module: Memory](#)
- [Module: Norm](#)
- [Module: Path](#)
- [Module: Random](#)
- [Module: RecordParser](#)
- [Module: Regexp](#)
- [Module: Search](#)

# Online Docs landing page (version 1.13)

A screenshot of the Chapel Documentation 1.13 landing page. The page has a dark sidebar on the left and a white main content area on the right. The sidebar contains a search bar and a navigation menu with sections like "COMPILING AND RUNNING CHAPEL", "WRITING CHAPEL PROGRAMS", and "LANGUAGE HISTORY". The main content area shows the "Chapel Documentation" page with sections for "Compiling and Running Chapel", "Writing Chapel Programs", "Language History", and an "Index". Each section has a list of links corresponding to the items in the sidebar.

Chapel Documentation 1.13

Search docs

COMPILING AND RUNNING CHAPEL

- Quickstart Instructions
- Using Chapel
- Platform-Specific Notes
- Technical Notes
- Tools

WRITING CHAPEL PROGRAMS

- Quick Reference
- Language Specification
- Built-in Types and Functions
- Standard Modules
- Package Modules
- Standard Layouts and Distributions
- Chapel Users Guide (WIP)

LANGUAGE HISTORY

- Chapel Evolution

Docs » Chapel Documentation

View page source

## Chapel Documentation

### Compiling and Running Chapel

- [Quickstart Instructions](#)
- [Using Chapel](#)
- [Platform-Specific Notes](#)
- [Technical Notes](#)
- [Tools](#)

### Writing Chapel Programs

- [Quick Reference](#)
- [Language Specification](#)
- [Built-in Types and Functions](#)
- [Standard Modules](#)
- [Package Modules](#)
- [Standard Layouts and Distributions](#)
- [Chapel Users Guide \(WIP\)](#)

### Language History

- [Chapel Evolution](#)

### Index

- [Chapel Online Documentation Index](#)

# Key Performance Optimizations (1.12 & 1.13)

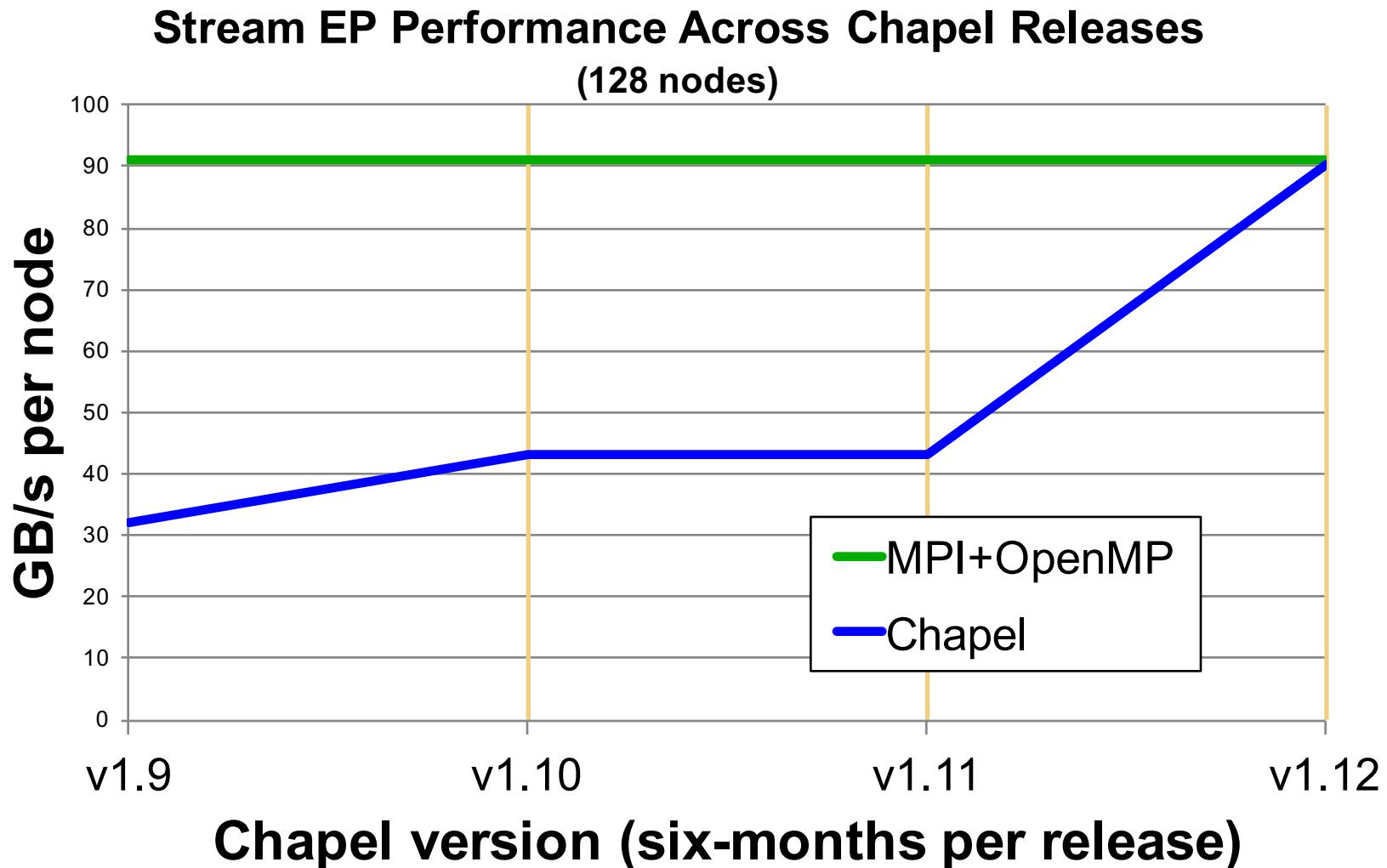
- **Compiler improvements:**

- locality and communication optimizations
- initial support for vectorization
- enabled competitive performance from LLVM back-end

- **Runtime improvements:**

- Qthreads-based tasking now used by default
- jemalloc-based memory now used by default
- ‘ugni’-based improvements for Cray networks

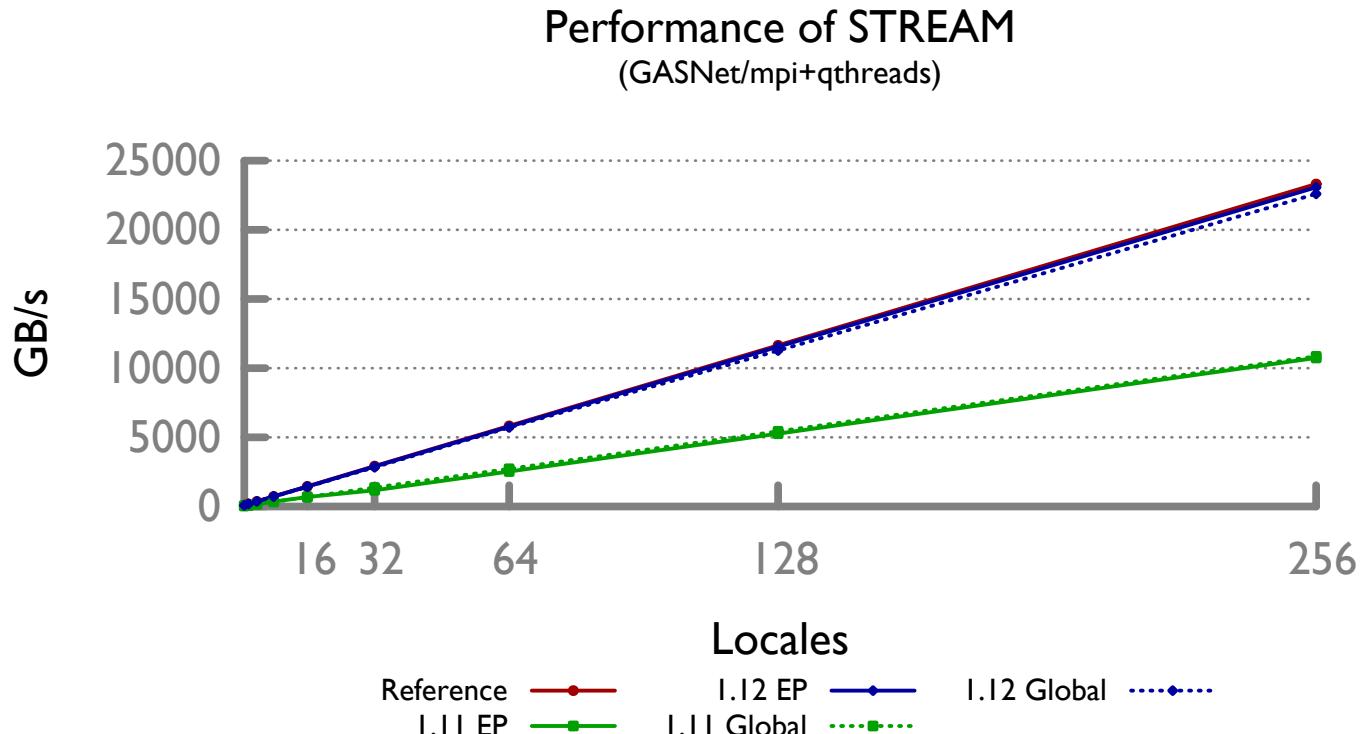
# Stream EP Performance (versions 1.9–1.12)



# STREAM Scaling (1.11 vs. 1.12, EP vs. global)

**Stream performance more than doubled from 1.11 to 1.12**

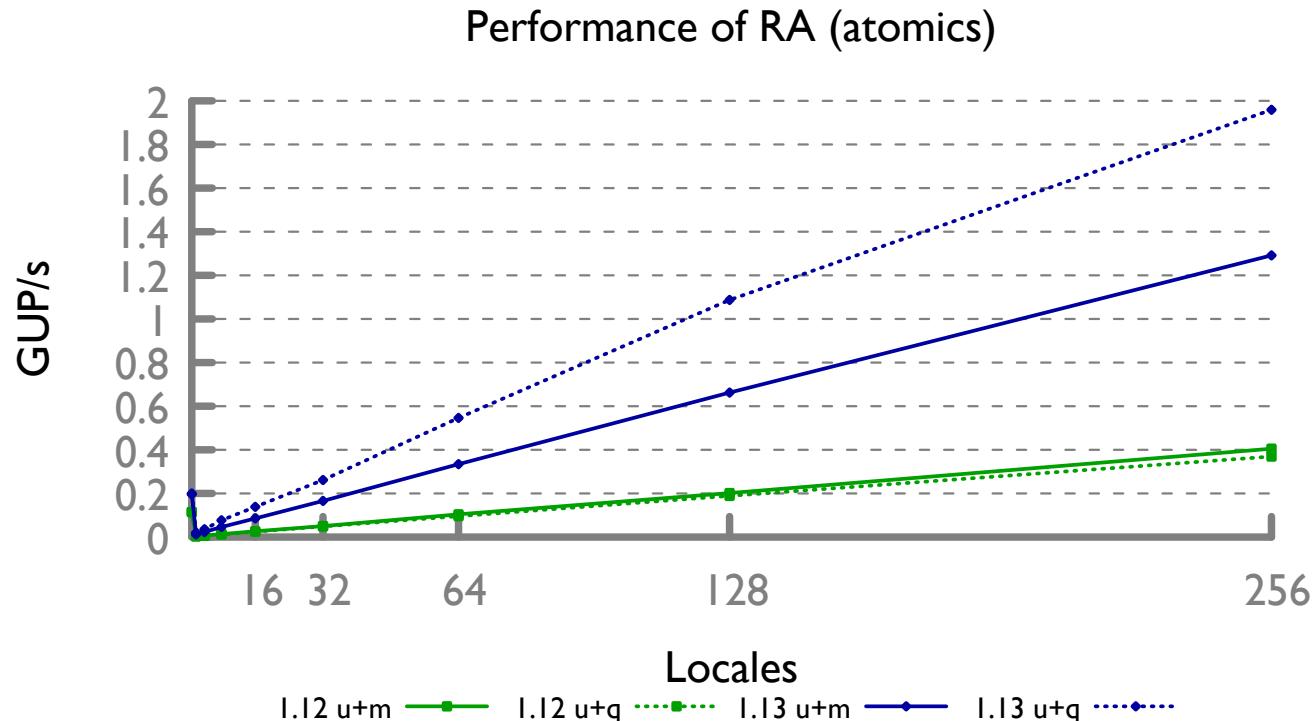
- EP is on par with the reference
- Global is also very competitive



# RA (atomics) Scaling (1.12 vs. 1.13, qthreads vs. muxed)

**RA perf (atomics version) improved from 1.12 to 1.13**

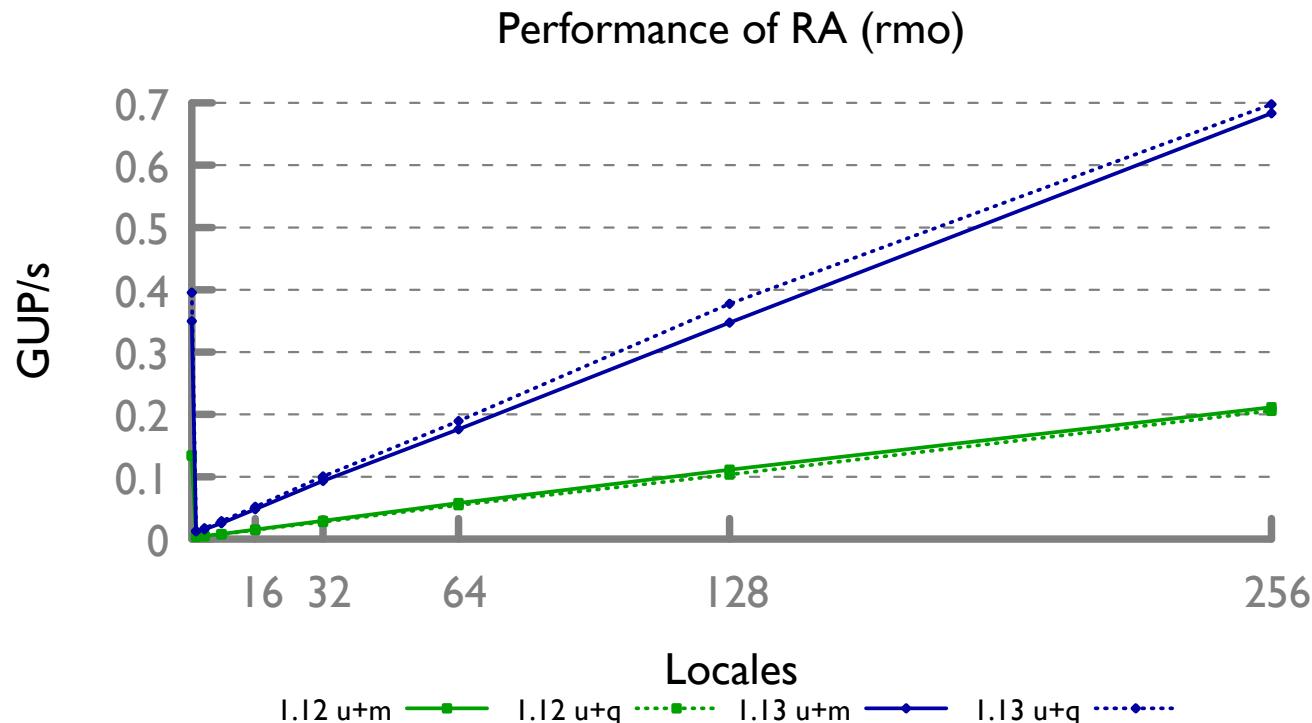
- ugni+qthreads performance improved 5x
- ugni+muxed performance improved 3x



# RA (rmo) Scaling (1.12 vs. 1.13, qthreads vs. muxed)

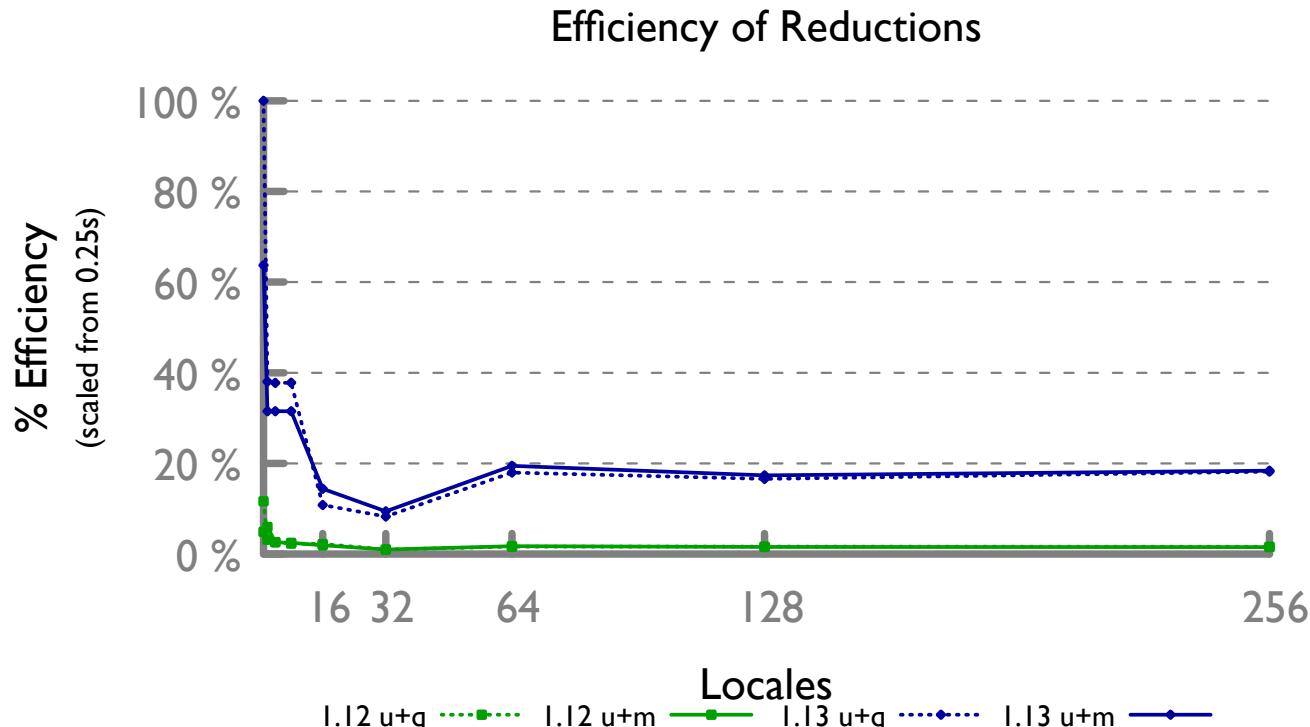
**RA (remote memory ops version) also improved for 1.13**

- 3x better performance for both ugni+qthreads and ugni+muxed



# Scaling of ‘reduce’ (1.12 vs. 1.13, qthreads vs. muxed)

- Reductions improved dramatically from 1.12 to 1.13
  - improved parallel efficiency/scalability
  - significantly improved raw performance



# Other Key Improvements (1.12 & 1.13)

- **Tools:**

- **chplvis**: visualize comm/tasking (see Phil Nelson's talk this afternoon)

- **Benchmarks:**

- **ISx**: Integer Sort proxy app (see Ben Harshbarger's talk this morning)
  - **LCALS**: Loop Kernels proxy app (see David Iten's talk this morning)

- **Portability:**

- **NUMA nodes**: significant locality improvements
  - **Intel® Xeon Phi™**: initial support (formerly “Knight’s Landing” / “KNL”)

- **Design Efforts:**

- initializers (constructor redesign)
  - error-handling
  - **CHIPs**: Chapel Improvement Proposals
    - see: <https://github.com/chapel-lang/chapel/blob/master/doc/chips/1.rst>

# Next Steps: Top Ten Priorities for 1.14

1. Close all remaining memory leaks and lock into testing
2. Finalize and implement initializers and copy semantics
3. Multi-locale performance improvements
4. NUMA/KNL locale models and performance (including HBM support)
5. Sparse / associative array improvements (distributed, performance)
6. Data analytics case studies
7. Single-locale performance improvements
8. Implement error-handling
9. Add comm/comp overlap for ‘qthreads’ (and retire ‘muxed’?)
10. Improve support for first-class functions / closures



# Chapel in the Community



COMPUTE

| STORE

| ANALYZE

# Computer Language Benchmark Game (CLBG)

## The CLBG recently started accepting Chapel entries

- Compares execution time, source size, memory, ... x 13 benchmarks
- <http://benchmarksgame.alioth.debian.org/>

The Computer Language Benchmarks Game

64-bit quad core data set

Will your toy benchmark program be faster if you write it in a different programming language? It depends how you write it!

Which programs are faster?

Which programs are smaller? Which programs use less CPU?

Ada	C	Clojure	C# Mono	C++	Dart
Erlang	F# Mono	Fortran	Go	Hack	
Haskell	Java	JavaScript	Lisp	Lua	
OCaml	Pascal	Perl	PHP	Python	
Racket	Ruby	JRuby	Rust	Scala	
Smalltalk	Swift	TypeScript			

Waiting until we've submitted more benchmarks before adding Chapel to front page

Racket	Ruby	JRuby	Rust	Scala
Smalltalk	Swift	TypeScript		
<b>We want easy answers</b> , but easy answers are often incomplete or wrong. You and I know, there's more we should understand:				
stories	details	fastest?		
conclusions				

# Chapel CLBG: entries to date

- Trying to review and submit 1–3 benchmarks per week
- Four entries submitted so far:
  - pidigits, n-body, mandelbrot, thread-ring
- Striving for a combination of elegance and performance
  - more heroic versions possible, but we're avoiding for first drafts
    - e.g., manually unrolling mandelbrot => 40% performance improvement



# Chapel CLBG: how are we doing?

**Noting that easy answers are often incomplete or wrong...**  
...we have three #1 entries so far: one for performance, two for size

The Computer Language Benchmarks Game						
<b>pidigits</b>						
description						
program source code, command-line and measurements						
x	source	secs	KB	gz		
1.0	<b>Chapel</b>	<b>1.60</b>	22,612	501		
1.1	Pascal Free Pascal #3	<b>1.73</b>	2,300	482		
1.1	C gcc	<b>1.73</b>	1,992	448		
1.1	Rust	<b>1.76</b>	6,372	1420		
1.2	Fortran Intel #3	<b>1.92</b>	1,572	975		
1.2	Lua #5	<b>1.94</b>	3,156	479		
1.4	Racket #2	<b>2.18</b>	21,312	1122		
1.4	PHP #5	<b>2.19</b>	8,852	394		
1.4	PHP #4	2.20	8,764	384		
1.4	Python 3 #5	<b>2.20</b>	10,248	335		
<b>thread-ring</b>						
description						
program source code, command-line and measurements						
x	source	secs	KB	gz		
1.0	<b>Chapel</b>	150.46	924,192	<b>209</b>		
1.0	Ruby #2	20 min	31,352	<b>215</b>		
1.3	Racket	151.46	115,152	<b>262</b>		
1.3	Erlang HiPE	41.61	21,720	<b>273</b>		
1.3	Erlang HiPE #2	41.59	32,548	273		
1.3	Erlang	39.50	21,356	<b>273</b>		
1.3	Erlang #2	39.68	23,912	273		
1.3	OCaml	7 min	27,836	<b>282</b>		
1.4	Python 3 #2	6 min	12,468	<b>288</b>		
1.4	OCaml #3	193.99	26,976	296		
<b>n-body</b>						
description						
program source code, command-line and measurements						
x	source	secs	KB	gz	cpu	cpu load
1.0	<b>Chapel</b>	23.26	20,652	<b>962</b>	23.26	100% 0% 1% 1%
1.1	Hack #3	26 min	133,128	<b>1080</b>	26 min	0% 15% 86% 1%
1.1	PHP #3	7 min	7,928	<b>1082</b>	7 min	0% 100% 0% 1%
1.2	Swift #4	40.86	4,736	<b>1129</b>	40.85	100% 1% 1% 1%
1.2	Swift	40.86	4,820	1135	40.84	1% 0% 1% 100%
1.2	Ruby #2	11 min	9,144	<b>1137</b>	11 min	0% 1% 100% 0%
1.2	Ruby JRuby #2	292.96	706,256	<b>1137</b>	5 min	26% 29% 25% 24%
1.2	Fortran Intel #4	21.91	512	<b>1172</b>	21.91	1% 0% 0% 100%
1.2	C gcc	21.15	952	<b>1173</b>	21.14	0% 1% 1% 100%
1.2	C gcc #6	21.40	1,024	1180	21.40	1% 100% 0% 1%

Interesting to compare execution times of similarly compact (non-heroic?) versions

# CLBG: Next Steps

- Complete initial round of submissions
- Tackle opportunities for improvement...

...without losing sight of scalability / distributed memory goals

...prioritizing ones that would benefit end-users of the language  
(don't want to optimize simply for CLBG's sake)



# Chapel's Google Summer of Code (GSoC)



- Applied to be a GSoC 2016 organization

- Compiled a project ideas list and developed new web content:  
<http://chapel.cray.com/gsoc>
- Were one of 178 accepted organizations from a pool of 369

A screenshot of the Chapel website. At the top, there's a navigation bar with a sun icon and the text "Google Summer of Code". The main content area has a dark blue header with the word "Chapel" and the URL "http://chapel.cray.com". Below this, there's a section titled "a Productive Parallel Programming Language" with a "About Chapel" link. A detailed paragraph describes Chapel as an open-source language for parallel computing. To the right, a modal window is open, showing the Cray logo and a "VIEW IDEAS LIST" button. It also contains a clock icon and text about announced projects, a "Technologies" section with buttons for "python", "chapel", "c", "c++", and "high performance computing".

Chapel

http://chapel.cray.com

a Productive Parallel Programming Language

About Chapel

Chapel is an [open-source](#) programming language designed for productive parallel computing at scale. Chapel is implemented with portability in mind, permitting Chapel to run on multicore desktops and laptops, commodity clusters, and the cloud, in addition to the high-end supercomputers for which it was designed. Chapel's design and development are being led by Cray Inc. in collaboration with academia, computing centers, and industry. Chapel offers a unique experience for students to work on projects involving high-performance computing, parallel programming, and compiler development.

VIEW IDEAS LIST

Accepted projects will be announced on April 22, 2016

Technologies

python chapel c  
c++  
high performance computing



COMPUTE

STORE

ANALYZE

# GSoC 2016: Submissions

- **Once announced, saw a sharp uptick in Chapel activity**
  - #chapel-developers chatroom; mailing lists; GitHub repos, forks, PRs
  - 3 GSoC applicants listed among contributors to version 1.13
- **46 student applications received**
  - granted 2 student slots from Google
  - would have liked to have taken many more if we'd been able
- **Some of the proposed projects:**
  - Stack-trace reporting on halts (accepted)
  - Incremental recompilation (accepted)
  - Online compiler for Chapel
  - Chapel GUI library
  - Begin expressions
  - STL-like data structures and algorithms
  - Build a web server in Chapel

# Chapel Blog Posts since CHIUW 2015



## Chapel Springs into a Summer of Code, April 2016

- highlighted 1.13 release, GSoC, CHIUW program, new Twitter feed

## Six Ways to Say “Hello” in Chapel (parts 1, 2, 3), Sept-Oct 2015

- overview of data- and task-parallel hello worlds for single-/multi-locale

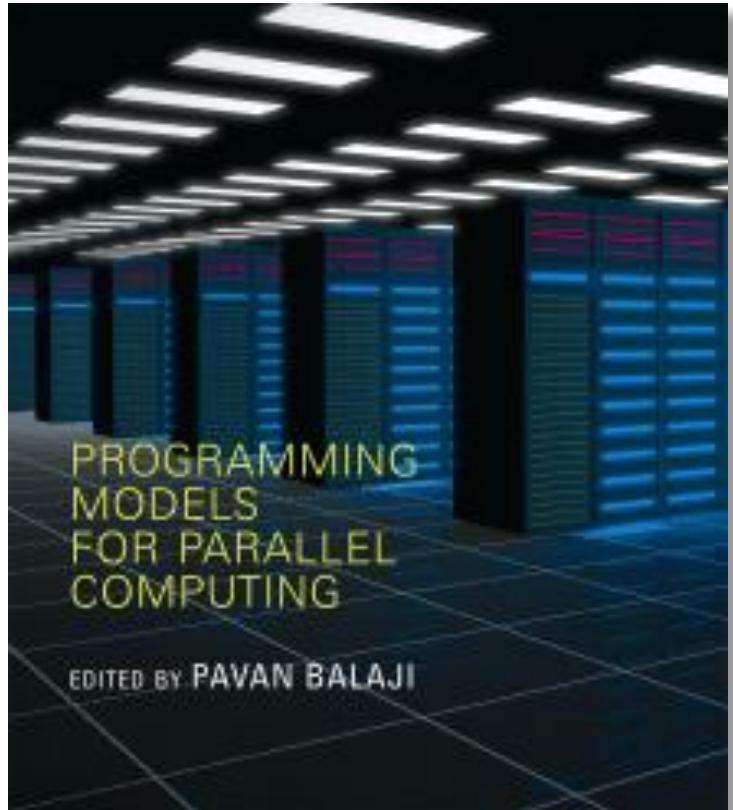
## Chapel Users ‘CHIUW’ Their Way Through Portland, June 2015

- overview of CHIUW 2015

# Chapel Book Chapter Published

Chapel chapter in ***Programming Models for Parallel Computing***

- a detailed overview of Chapel's history, motivating themes, features
- edited by Pavan Balaji, published by MIT Press, November 2015
- chapter is now also available [online](#)





# Chapel added to GitHub PL Showcase (Nov 2015)

Screenshot of the GitHub PL Showcase page for Programming languages.

The page shows a list of active programming languages on GitHub, currently ranked #32/35. The languages listed are:

- apple/swift** (Swift) - C++ ★ 29,755
- rust-lang/rust** (Rust) - Rust ★ 16,451
- golang/go** (Go) - Go ★ 16,370
- dart-lang/sdk** (Dart) - Dart ★ 635 ⚡ 89
- chapel-lang/chapel** (Chapel) - Chapel ★ 256 ⚡ 100
- dylan-lang/opendylan** (Dylan) - Dylan ★ 212 ⚡ 49

A "View all" button is located at the bottom right of the showcase list.



COMPUTE

| STORE

| ANALYZE

# Chapel AMA on /r/ProgrammingLanguages

- Held Wednesday Oct 14, 2016 as an all-day session
  - Cray Chapel team drafted and edited answers
  - followed up on straggling questions over the following two days

This is an archived post. You won't be able to vote or comment.

**[AMA] We're the development team for the Chapel parallel programming language, Ask Us Anything!** (self.ProgrammingLanguages)

submitted 7 months ago \* by Chapel\_team\_at\_Cray

Thanks to everyone who participated in the AMA this week -- for us, it was a lively and interesting session. If you have further questions about Chapel, please check out <http://chapel.cray.com> and don't hesitate to contact us through the [community mailing lists](#) or by mailing the Chapel team at Cray at [chapel\\_info@cray.com](mailto:chapel_info@cray.com) (removing the spaces).

Parallel computers are notoriously difficult to program, particularly at large scales. Chapel is an open-source programming language that we are developing to address this challenge. Specifically, Chapel is designed to simplify the creation of parallelism and management of locality using a modern and productive language design.

Chapel's design and implementation have been undertaken with portability in mind, permitting its programs to run on parallel systems of all scales, from multicore desktops and laptops, to commodity clusters and the cloud, along with the high-end supercomputers for which it was designed. Our team leads the design and development of Chapel, in collaboration with members of academia, computing centers, and industry in the U.S. and around the world.

To give a trivial taste of Chapel, the following program distributes its parallel loop's iterations across all the processor cores of a distributed memory system to print "Hello world!" style messages in parallel:

```
config const n = 1000;
use CyclicDist;
const ProblemSpace = {1..n} dmapped Cyclic(startIdx=1);
forall i in ProblemSpace do
  writeln("Hello from iter #", i, " running on node ", here.id);
```

Today's AMA is hosted by the Chapel development team at Cray Inc.:

- Brad Chamberlain, technical lead
- Tom MacDonald, project manager
- Ben Albrecht
- Kyle Brady
- Lydia Duncan

Want to join? Log in or sign up in seconds. | English

search

this post was submitted on 14 Oct 2015  
**27 points** (74% upvoted)  
shortlink: <https://redd.it/3oqalv>

username password  
 remember me [reset password](#) [login](#)

/r/DesiWeddings

discuss this ad on reddit

[Submit a new link](#)

[Submit a new text post](#)

**ProgrammingLanguages**

[subscribe](#) | 3,853 readers  
● ~4 users here now

**Welcome!**

This subreddit is dedicated to discussion of programming languages, their theory, syntax and compilers. Post your ideas and get constructive criticism.  
Be nice, contribute, and stay away from useless flame wars.



COMPUTE | STORE | ANALYZE

# Chapel AMA on /r/ProgrammingLanguages

- Thread received 58 comments + 34 responses overall
  - 24 unique user IDs, 19 of whom posted top-level questions
- Received a net score of 26 points, 75% upvoted
  - currently #2 in top links from the past year

 reddit PROGRAMMINGLANGUAGES hot new rising controversial top gilded promoted

links from: [past year](#)

1	33	"Screw it, I'll make my own!" - The story of a new programming language	(breuleux.net)
		submitted 3 months ago by <a href="#">MadcapJake</a>	
		<a href="#">16 comments</a> <a href="#">share</a>	
2	26	[AMA] We're the development team for the Chapel parallel programming language, Ask Us Anything!	
		submitted 7 months ago * by <a href="#">Chapel_team_at_Cray</a>	
		<a href="#">92 comments</a> <a href="#">share</a>	
3	26	The Language Strangeness Budget	(words.steveklabnik.com)
		submitted 7 months ago by <a href="#">WalkerCodeRanger</a>	
		<a href="#">comment</a> <a href="#">share</a>	
4	21	Pony: an ML-family programming language, a statically typed and compiled Erlang, or simply a LLVM-beat C/C++ speed by minimizing atomics and synchronization	(self.ProgrammingLanguages)
		submitted 24 days ago * by <a href="#">Zen_X</a>	
		<a href="#">8 comments</a> <a href="#">share</a>	
5	23	The Swift Programming Language's Most Commonly Rejected Change Requests	(github.com)
		submitted 4 months ago by <a href="#">WalkerCodeRanger</a>	
		<a href="#">comment</a> <a href="#">share</a>	



COMPUTE

| STORE

| ANALYZE

# Chapel at PGAS15 (Washington DC, Sept 16-18)

## **Productive Programming in Chapel: A Computation-Driven Introduction**, tutorial, Brad Chamberlain and Michael Ferguson, Sept 16

- 3-hour tutorial with 2 hands-on sessions
- dry-run of SC15 tutorial
- ~12 attendees (though some familiar faces in audience, some here today)

## **Five Things You Should Do to Create a Future-Proof Exascale Language**, invited talk, Brad Chamberlain, Washington DC, Sept 17

- 45-minute talk
- call-to-arms for PGAS language designers and why Chapel's on-track
- ~30-35 attendees, primarily PGAS students and researchers (same for following talks)

## **Caching Puts and Gets in a PGAS Language Runtime**, research paper and talk, Michael Ferguson, Sept 17

- 30-minute talk
- description of compiler optimization to cache fine-grained communication

## **Toward a Data-Centric Profiler for PGAS Applications [poster]**, research poster and talk, Hui Zhang (University of Maryland), Sept 17

- research poster and short talk
- analysis to tie execution time to data structures rather than control structures

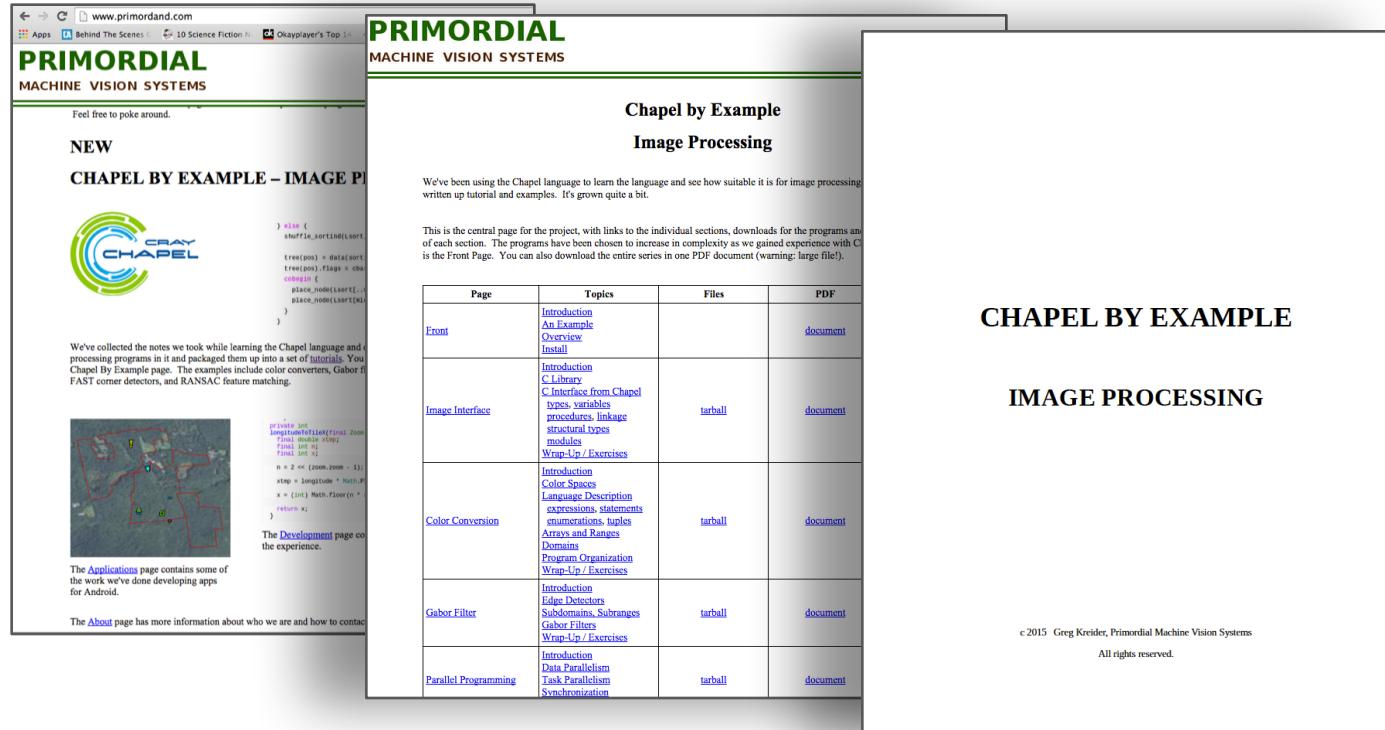


# Notable “Learning Chapel” Artifacts

Learn Chapel in Y Minutes, <http://learnxinyminutes.com>, Ian Bertolacci

- dense one-file overview of Chapel

Chapel By Example: Image Processing, manual, Greg Kreider, [Primordial Machine Vision Systems](#)



The screenshot shows two views of the 'Chapel by Example' website. On the left is a smaller preview of the homepage, which includes a 'PRIMORDIAL MACHINE VISION SYSTEMS' logo, a 'NEW' section, and a 'CHAPEL BY EXAMPLE – IMAGE PROCESSING' section. It also features a 'CRAY CHAPEL' logo and some sample code. On the right is the full homepage, titled 'Chapel by Example Image Processing'. It contains a brief introduction and a table of contents with links to various sections like 'Introduction', 'Color Spaces', 'Language Description', etc., each with a 'tarball' download link and a 'document' link.

Page	Topics	Files	PDF
Front	Introduction An Example Overview Install		document
Image Interface	Introduction C Library C Interface from Chapel types, variables procedures, linkage structural types modules Wrap-Up / Exercises	tarball	document
Color Conversion	Introduction Color Spaces Language Description expressions, statements enumerations, tuples Arrays and Ranges Domains Program Organization Wrap-Up / Exercises	tarball	document
Gabor Filter	Introduction Edge Detectors Subdomains, Subranges Gabor Filters Wrap-Up / Exercises	tarball	document
Parallel Programming	Introduction Data Parallelism Task Parallelism Synchronization	tarball	document

# Summary

- **Chapel is thriving**
  - significant improvements with each release
    - capabilities and performance
  - strong core team at Cray—largest and most capable ever
  - continued growth in the community and in outward-facing forums
- **CHIUW is a crucial part of Chapel's growth**
  - thanks to today's speakers and audience members for participating



# 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, and URIKA. 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.*





**CRAY**  
THE SUPERCOMPUTER COMPANY