State of the Chapel Project 2016

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

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)

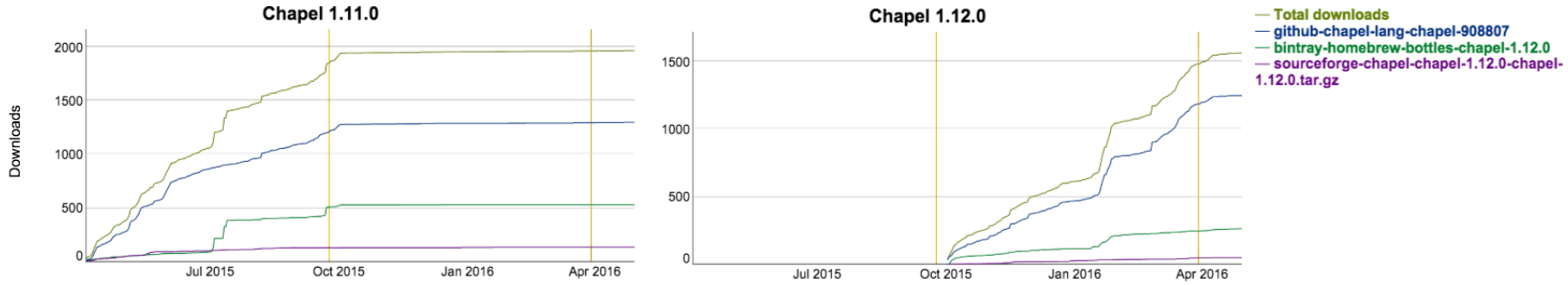
A Year in the Life of Chapel

- **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 CHIUW 2015

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

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

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

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

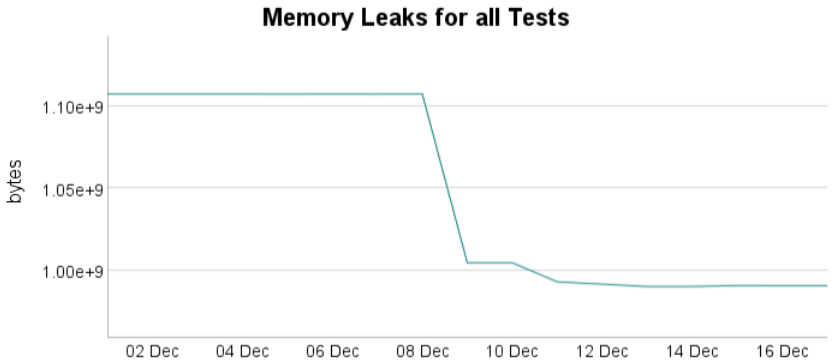
New string implementation was enabled on December 9th

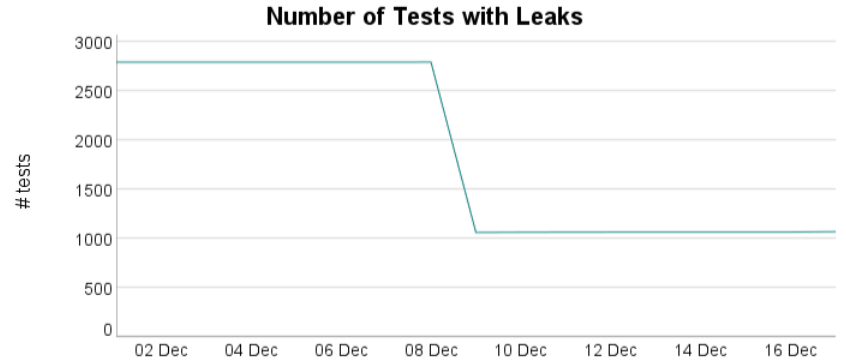
In our test suite…

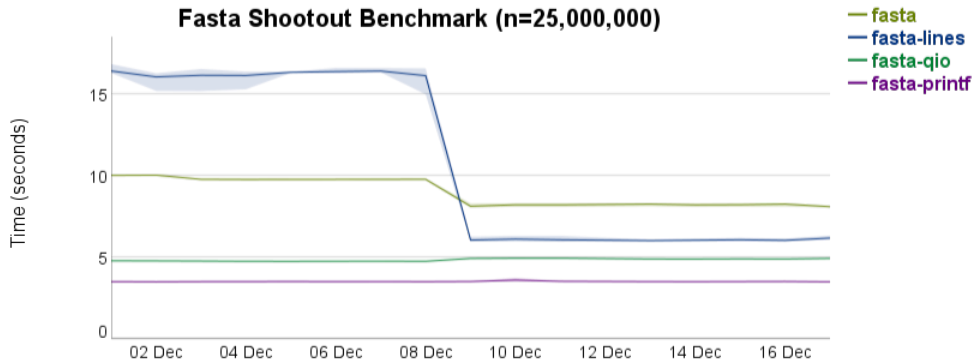
…string-related memory leaks went from 123MB to 22MB
- by 1.13, down to ~300B

…# 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()      // substring
these()     // iterate over chars
startsWith()
endsWith()
find()
rfind()
count()
replace()
split()
join()
strip()
partition()
localize()
c_str()
```

```
isEmptyString()
isUpper()
isLower()
isSpace()
isAlpha()
isDigit()
isAlnum()
isPrintable()
isTitle()
toLower()
toUpper()
toTitle()
capitalize()
+, +=, *, ==, !=, <=, ...
```

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

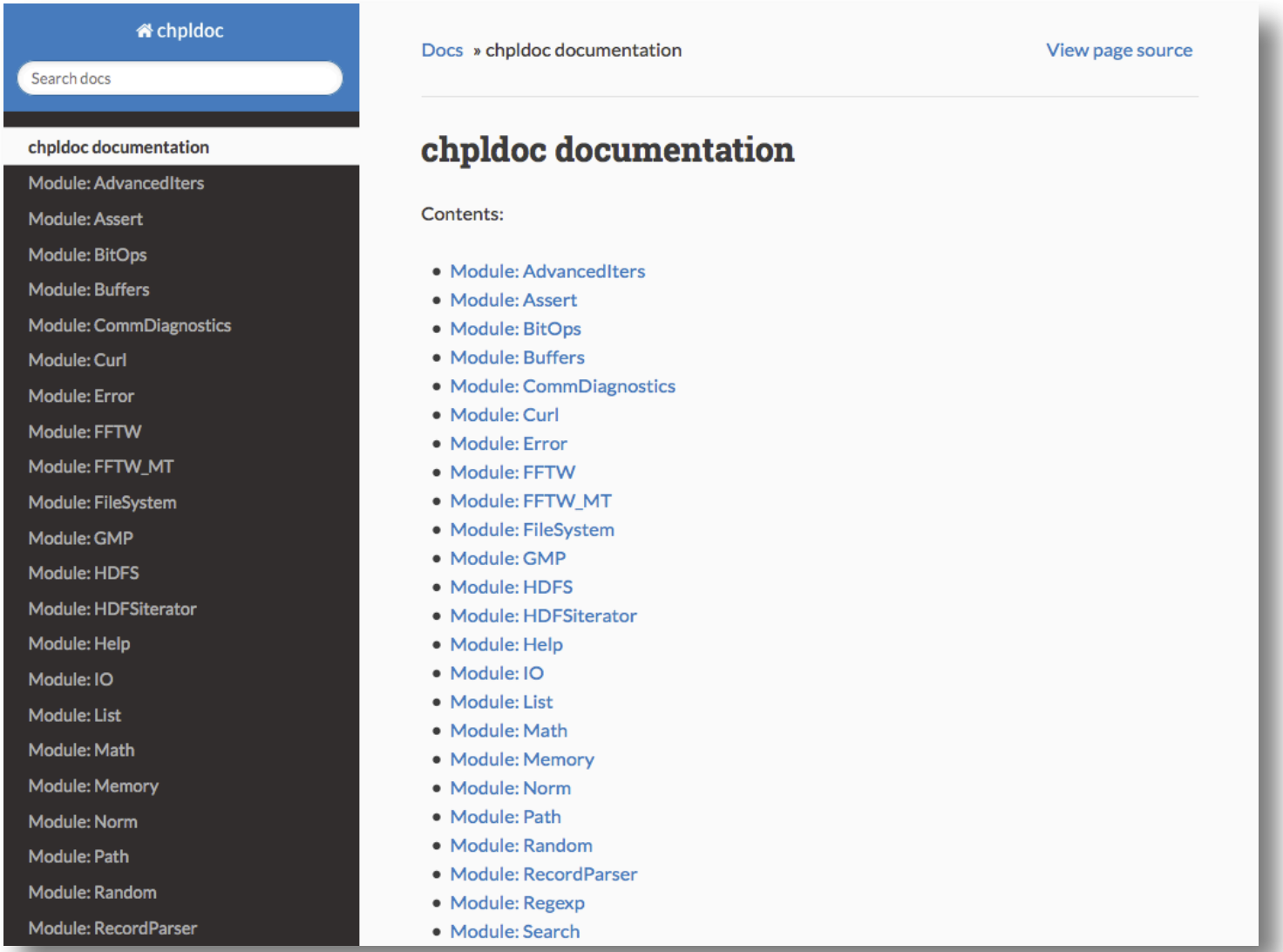
CHIUW 2016: Docs significantly expanded (125+ pages):

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

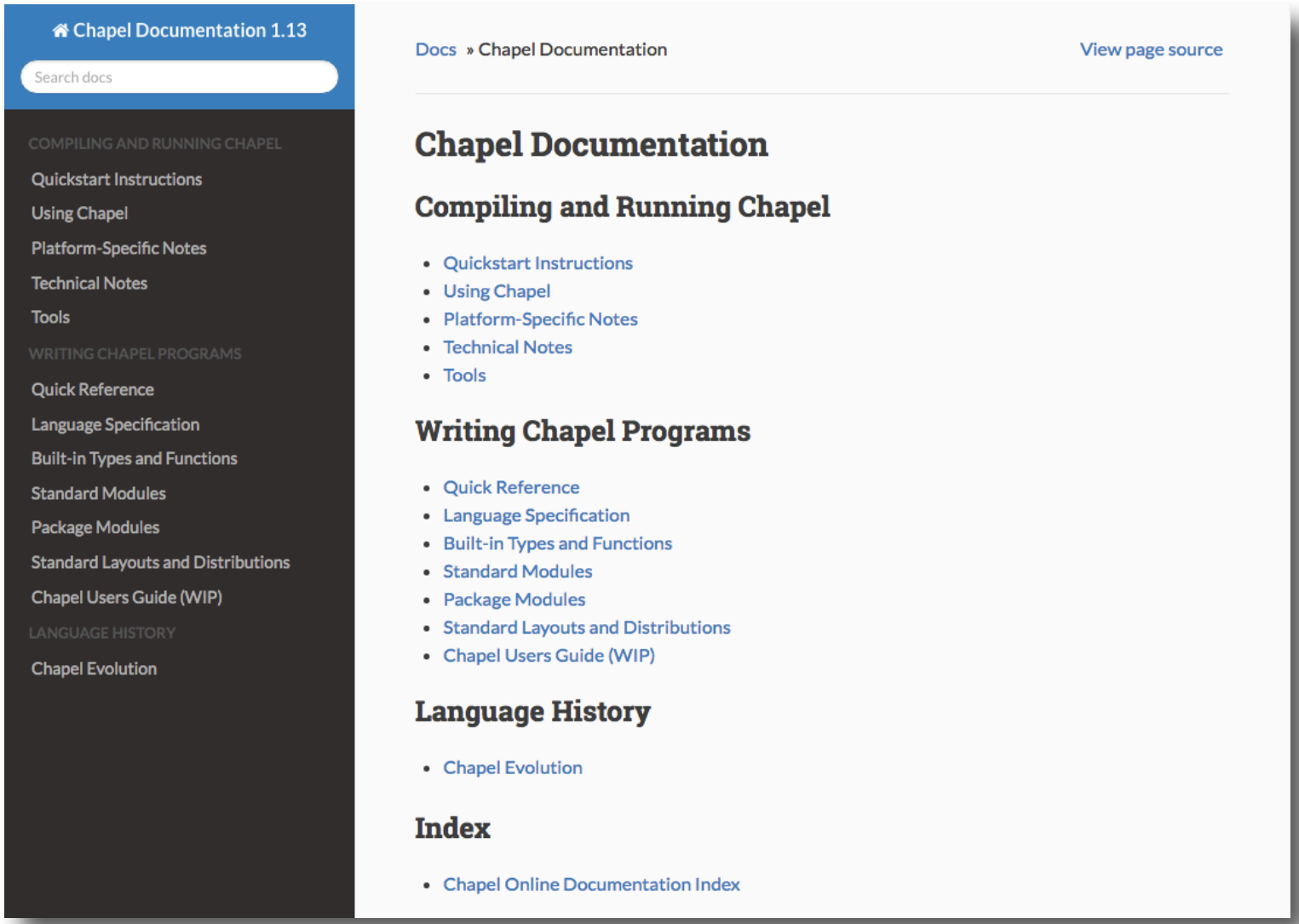
…also significantly reorganized for clarity

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

Online Docs landing page (version 1.11)



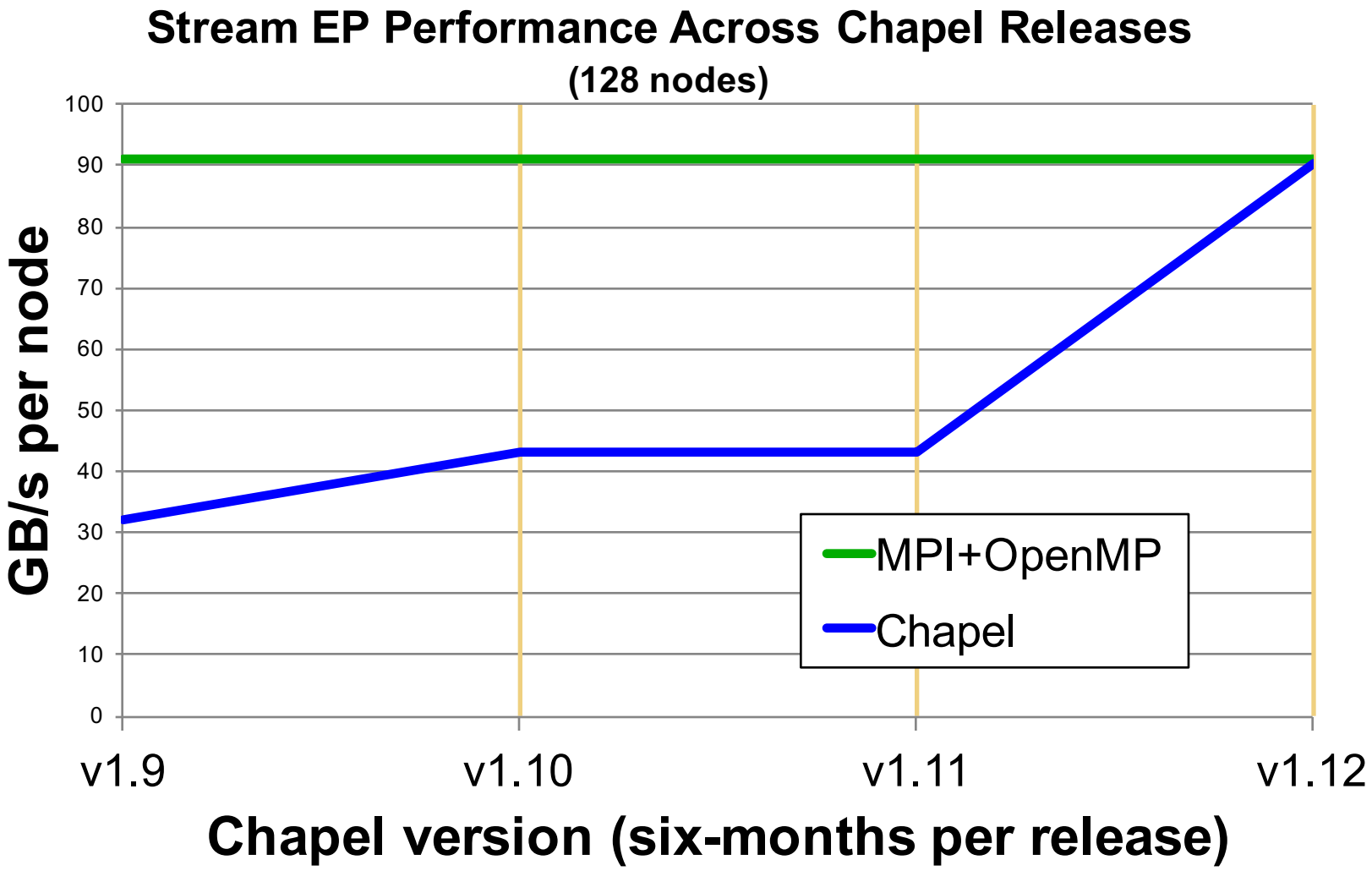
Online Docs landing page (version 1.13)



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

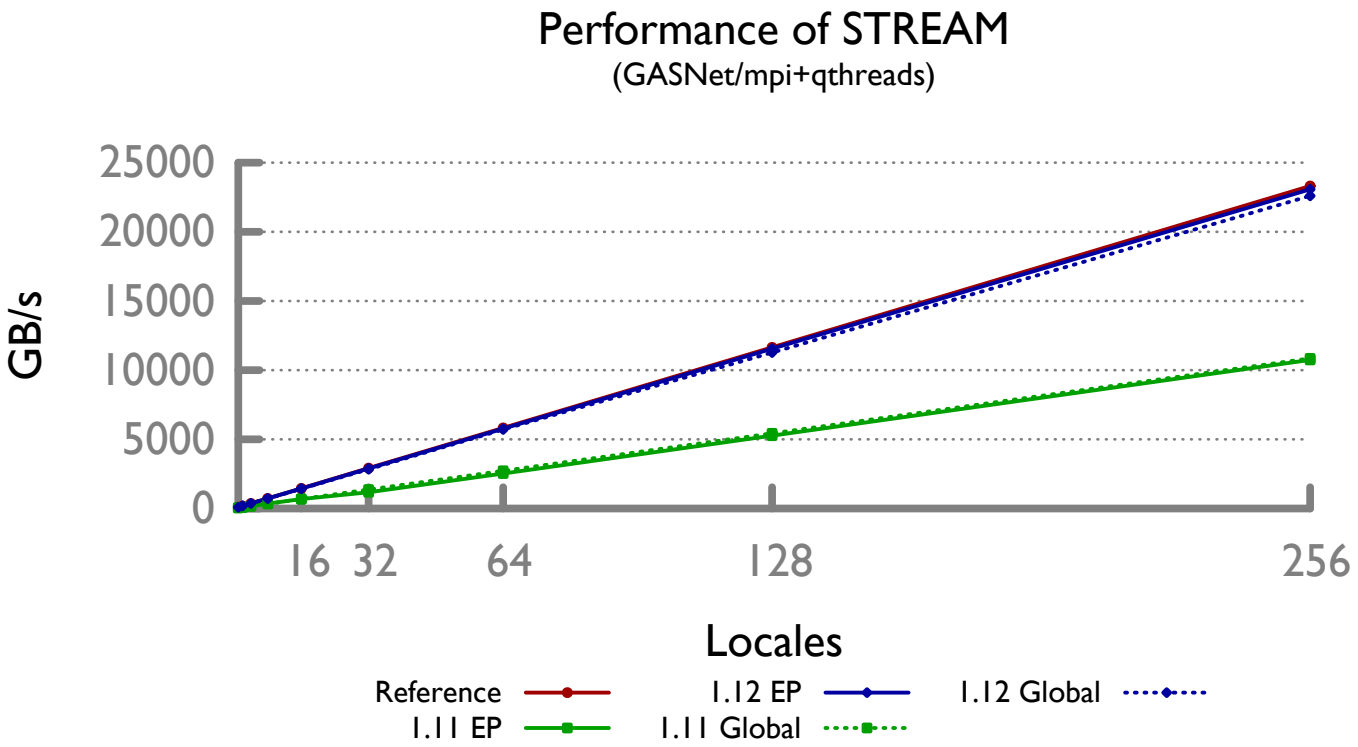
Stream EP Performance (versions 1.9–1.12)



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

Stream performance more than doubled from 1.11 to 1.12

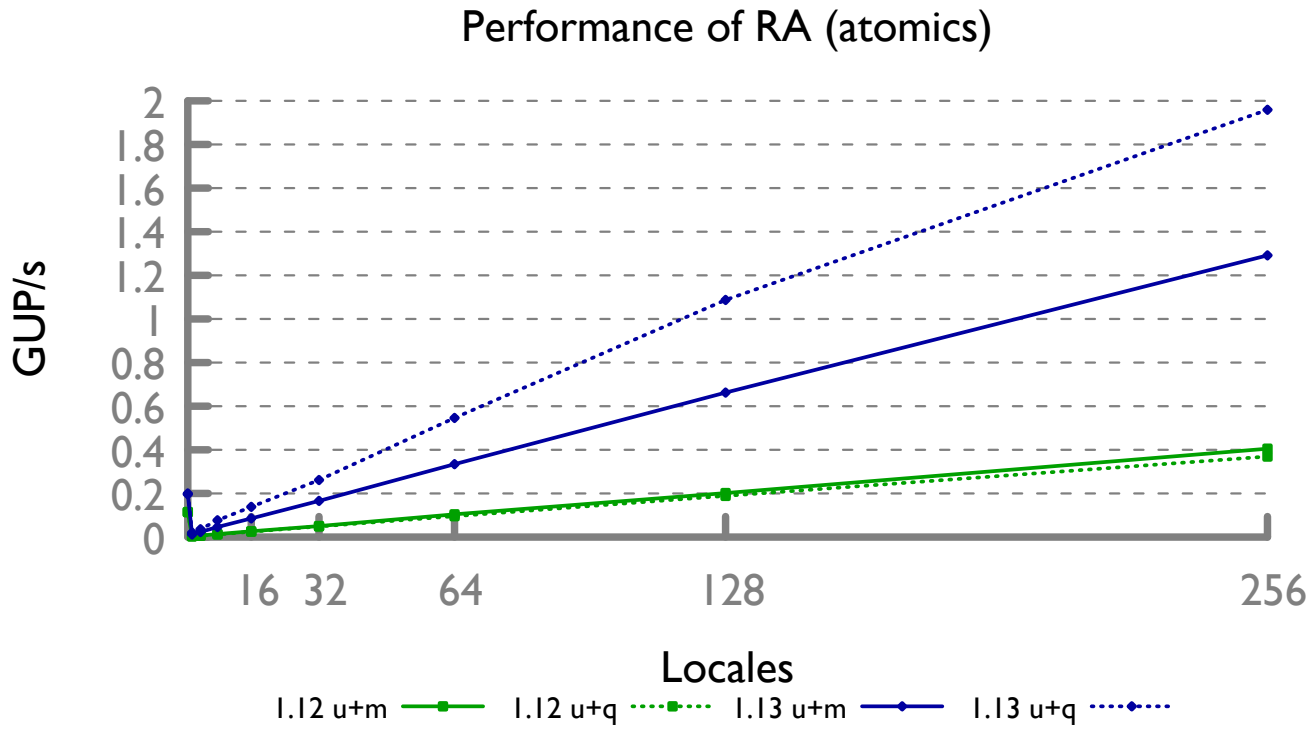
- EP is on par with the reference
- Global is also very competitive



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

RA perf (atomics version) improved from 1.12 to 1.13

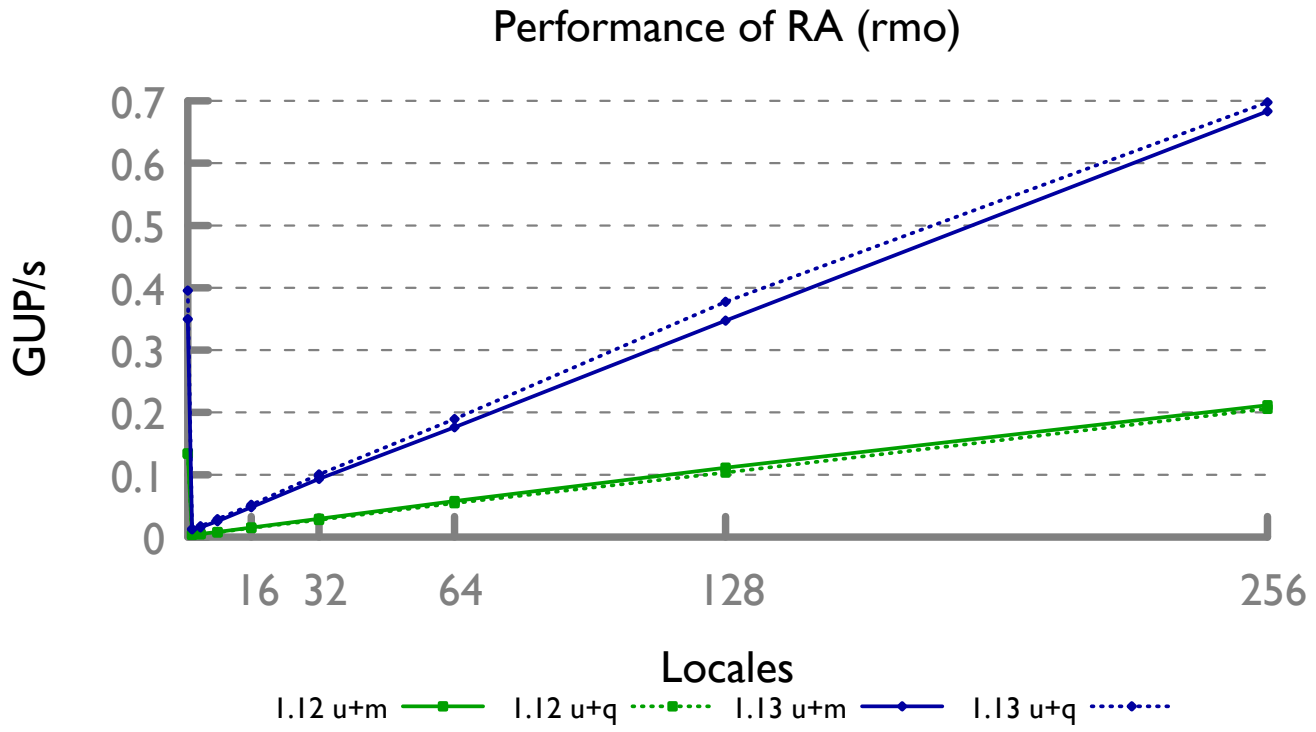
- ugni+qthreads performance improved 5x
- ugni+muxed performance improved 3x



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

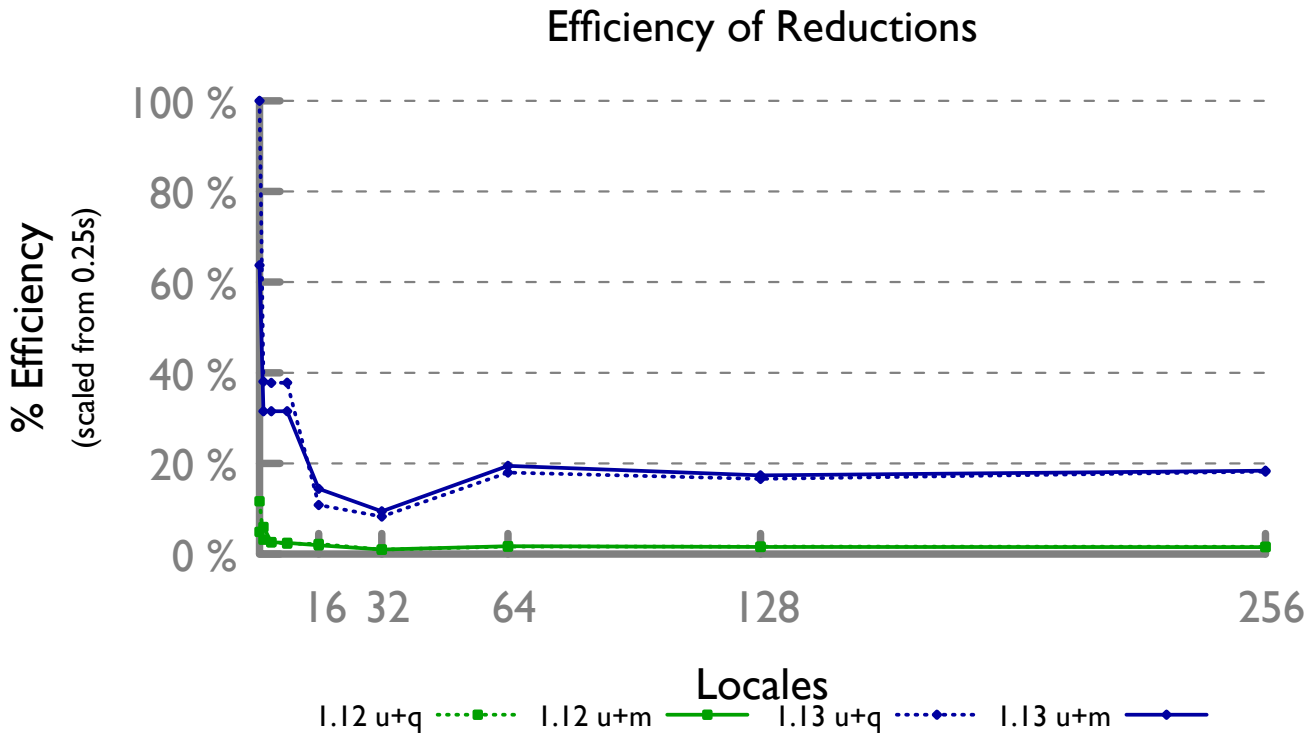
RA (remote memory ops version) also improved for 1.13

- 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

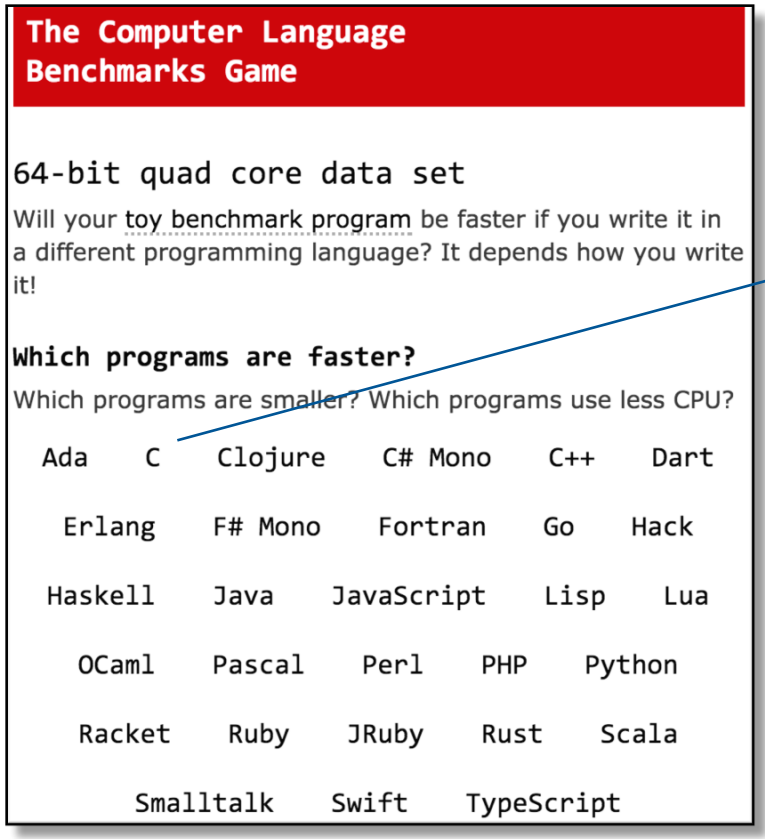
1. **Close all remaining memory leaks and lock into testing**
2. **Finalize and implement initializers and copy semantics**
3. **Multi-locale performance improvements**
4. **NUMA/KNL locale models and performance** (including HBM support)
5. **Sparse / associative array improvements** (distributed, performance)
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

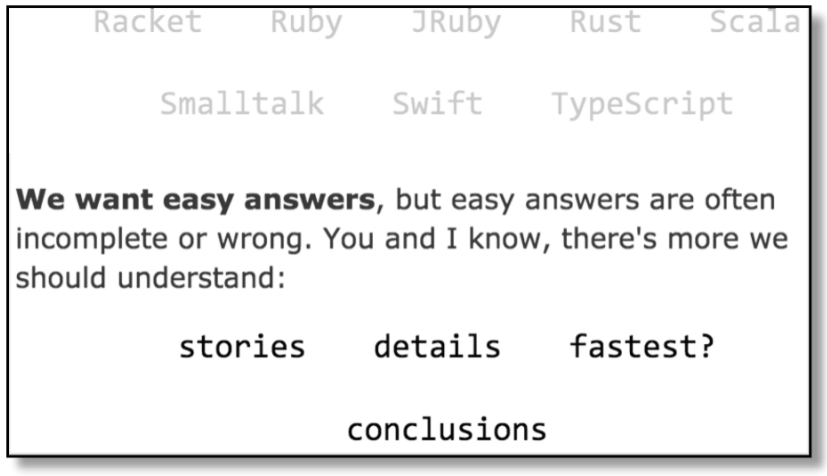
Computer Language Benchmark Game (CLBG)

The CLBG recently started accepting Chapel entries

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



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



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

pidigits

description

program source code, command-line and measurements

×	source	secs	KB	gz
1.0	**Chapel**	**1.60**	22,612	501
1.1	**Pascal** Free Pascal #3	**1.73**	2,300	482
1.1	**C** gcc	**1.73**	1,992	448
1.1	**Rust**	**1.76**	6,372	1420
1.2	**Fortran** Intel #3	**1.92**	1,572	975
1.2	**Lua** #5	**1.94**	3,156	479
1.4	**Racket** #2	**2.18**	21,312	1122
1.4	**PHP** #5	**2.19**	8,852	394
1.4	**PHP** #4	2.20	8,764	384
1.4	**Python 3** #5	**2.20**	10,248	335

The Computer Language Benchmarks Game

thread-ring

description

program source code, command-line a measurements

×	source	secs	KB	gz
1.0	**Chapel**	150.46	924,192	**209**
1.0	**Ruby** #2	20 min	31,352	**215**
1.3	**Racket**	151.46	115,152	**262**
1.3	**Erlang** HiPE	41.61	21,720	**273**
1.3	**Erlang** HiPE #2	41.59	32,548	273
1.3	Erlang	39.50	21,356	**273**
1.3	Erlang #2	39.68	23,912	273
1.3	**OCaml**	7 min	27,836	**282**
1.4	**Python 3** #2	6 min	12,468	**288**
1.4	**OCaml** #3	193.99	26,976	296

The Computer Language Benchmarks Game

n-body

description

program source code, command-line and measurements

×	source	secs	KB	gz	cpu	cpu load
1.0	**Chapel**	23.26	20,652	**962**	23.26	100% 0% 1% 1%
1.1	**Hack** #3	26 min	133,128	**1080**	26 min	0% 15% 86% 1%
1.1	**PHP** #3	7 min	7,928	**1082**	7 min	0% 100% 0% 1%
1.2	**Swift** #4	40.86	4,736	**1129**	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	**1137**	11 min	0% 1% 100% 0%
1.2	Ruby JRuby #2	292.96	706,256	**1137**	5 min	26% 29% 25% 24%
1.2	**Fortran** Intel #4	21.91	512	**1172**	21.91	1% 0% 0% 100%
1.2	**C** gcc	21.15	952	**1173**	21.14	0% 1% 1% 100%
1.2	**C** gcc #6	21.40	1,024	1180	21.40	1% 100% 0% 1%

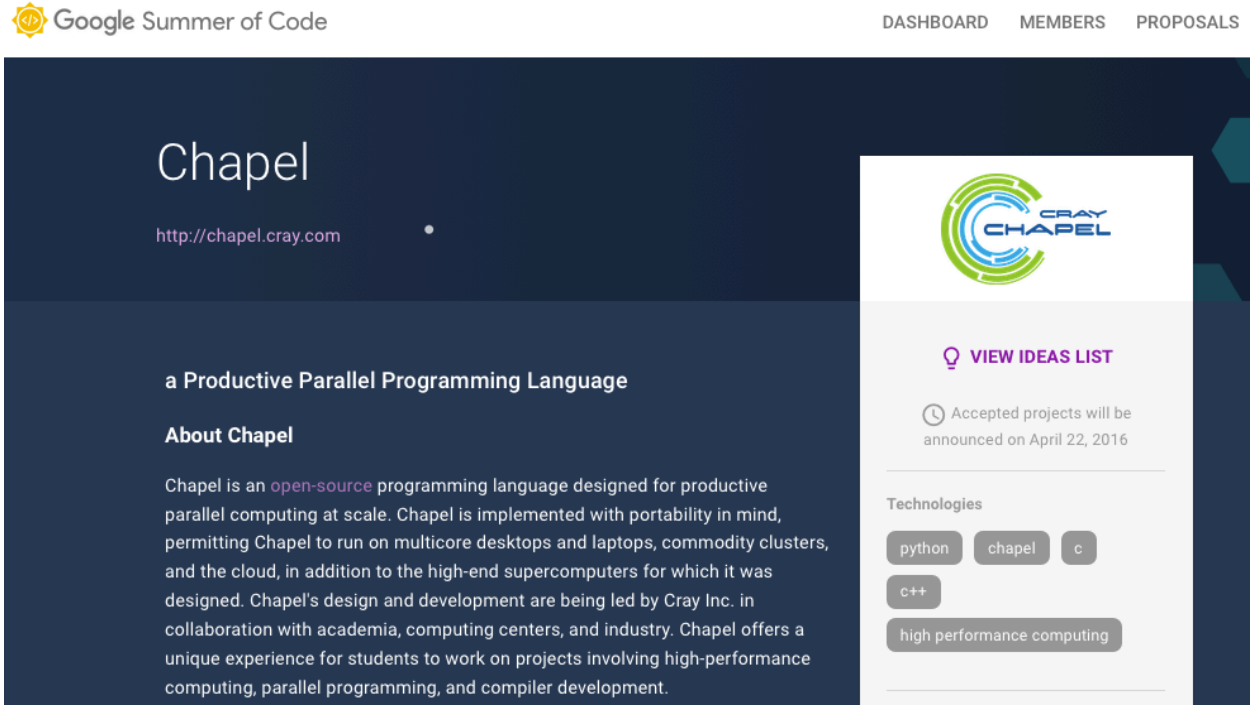
Interesting to compare execution times of similarly compact (non-heroic?) versions

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)

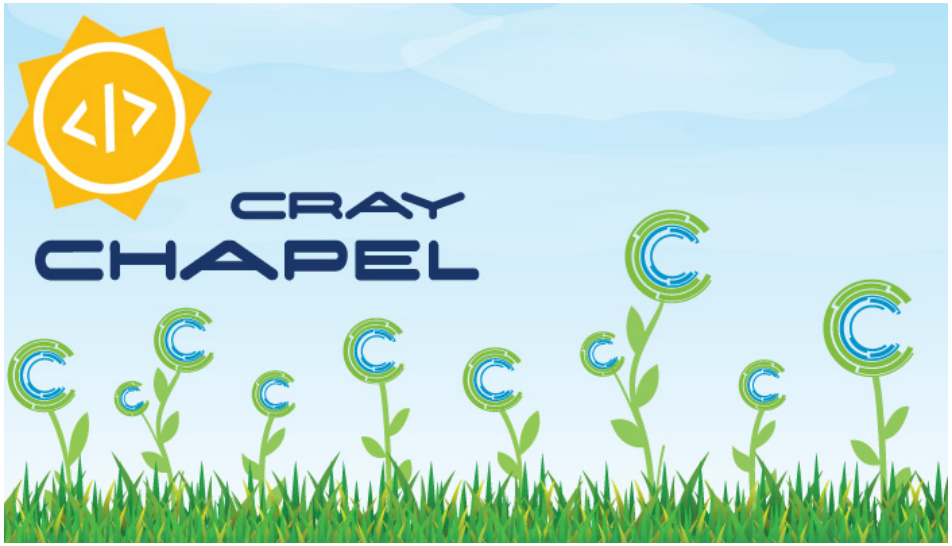
- **Applied to be a GSoC 2016 organization**
  - Compiled a project ideas list and developed new web content: http://chapel.cray.com/gsoc
  - Were one of 178 accepted organizations from a pool of 369



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

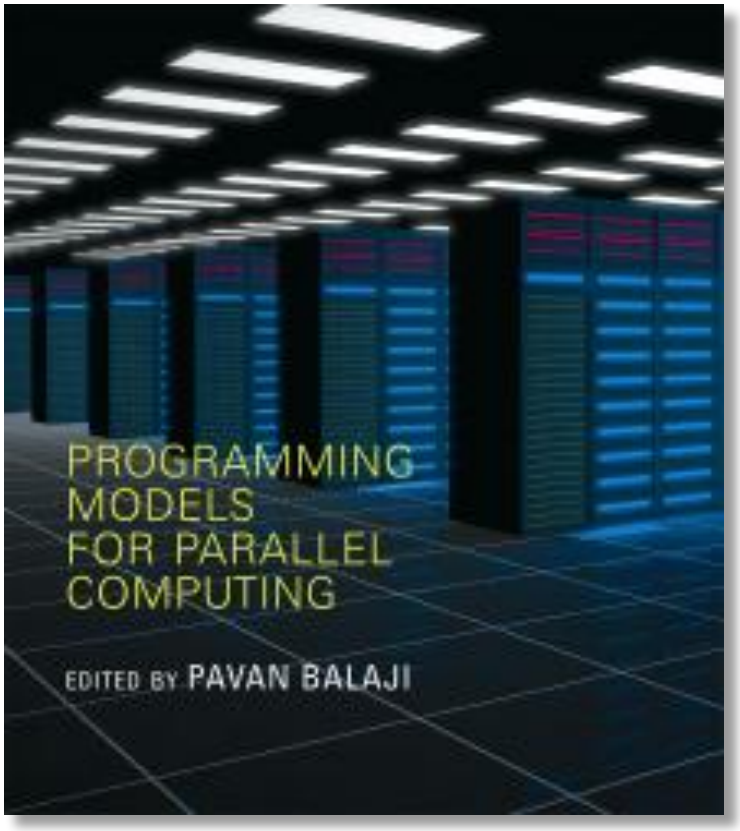
Chapel Users `CHIUW' Their Way Through Portland, June 2015

- overview of CHIUW 2015

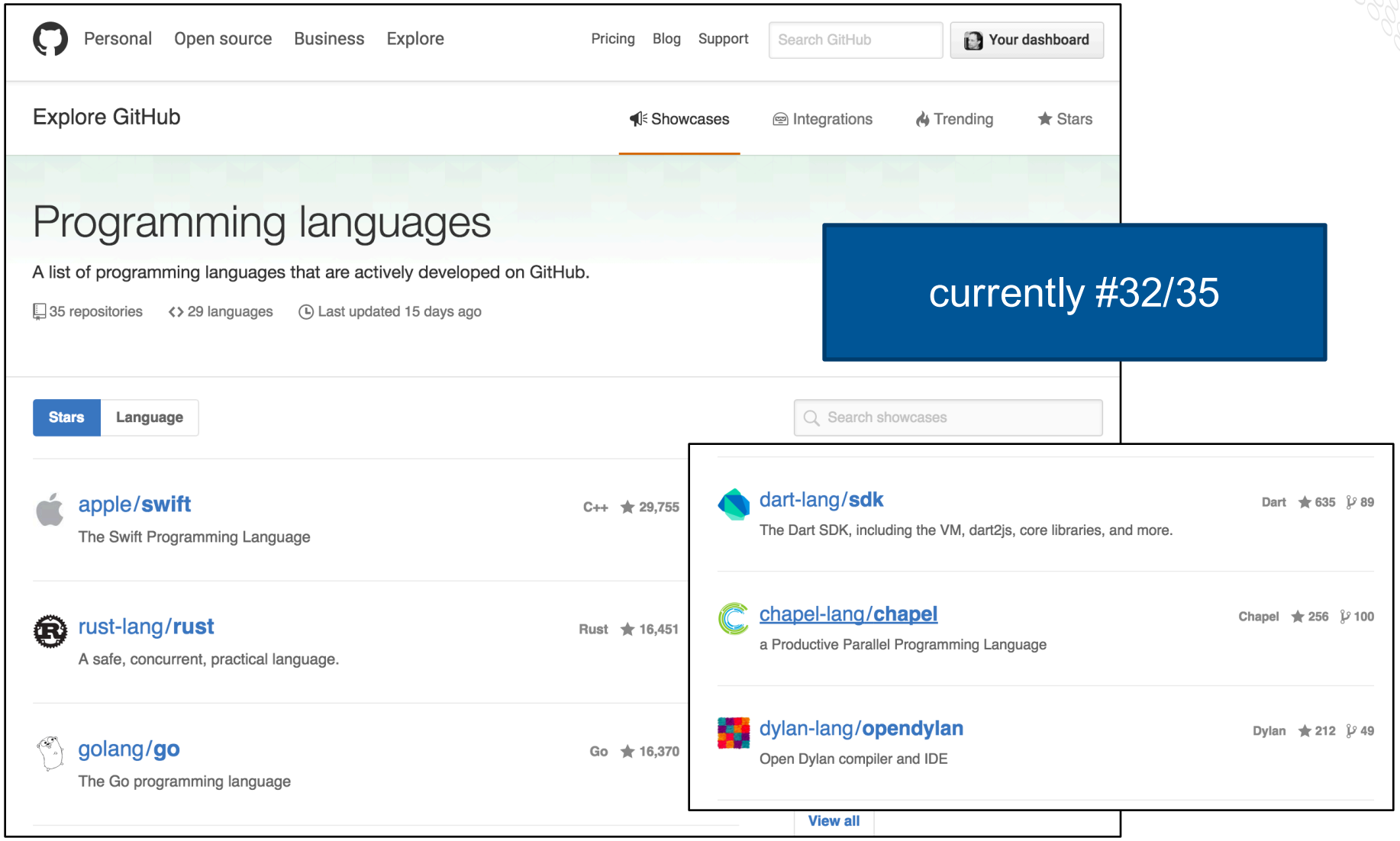
Chapel Book Chapter Published

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

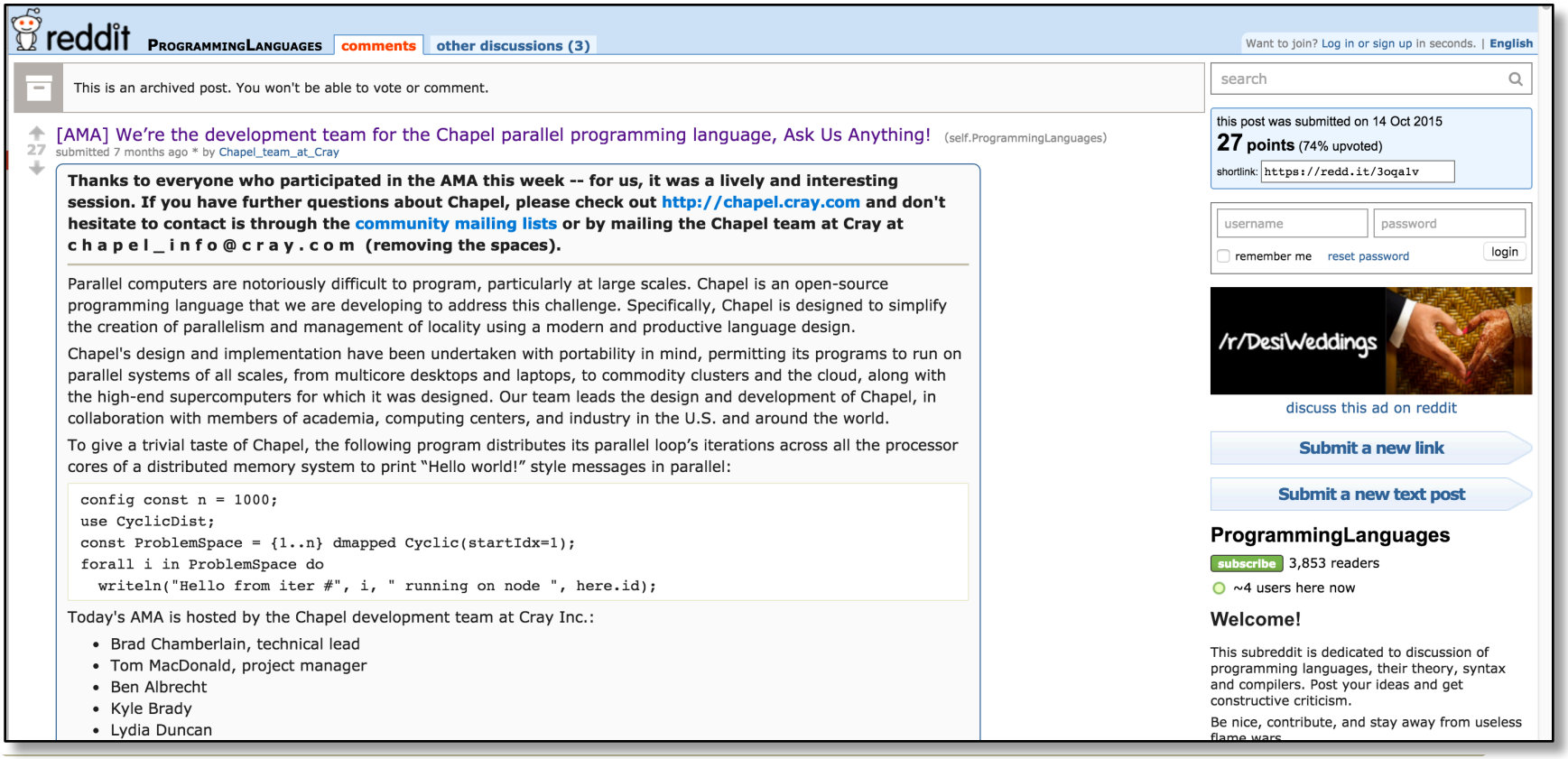


Chapel added to GitHub PL Showcase (Nov 2015)



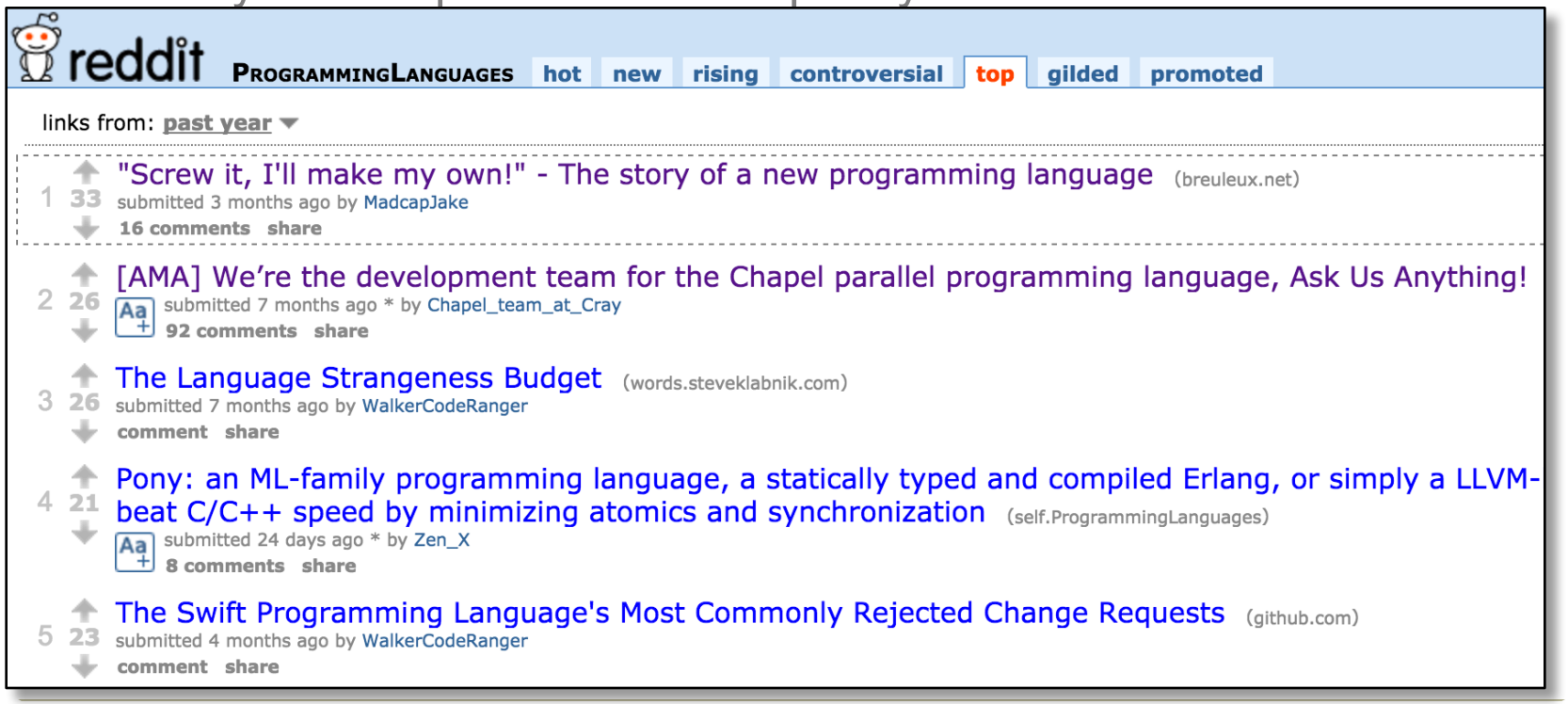
Chapel AMA on /r/ProgrammingLanguages

- **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



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



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

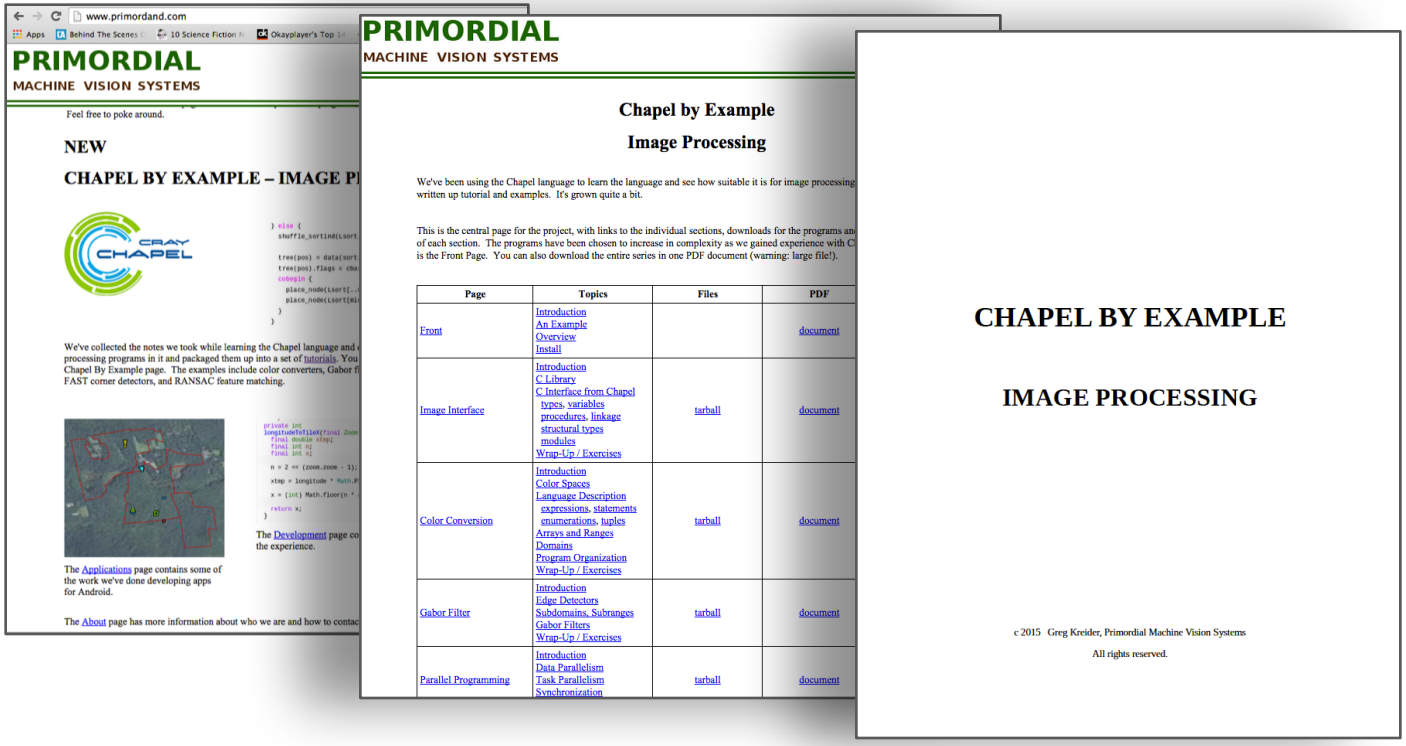
- research poster and short talk
- analysis to tie execution time to data structures rather than control structures

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



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.

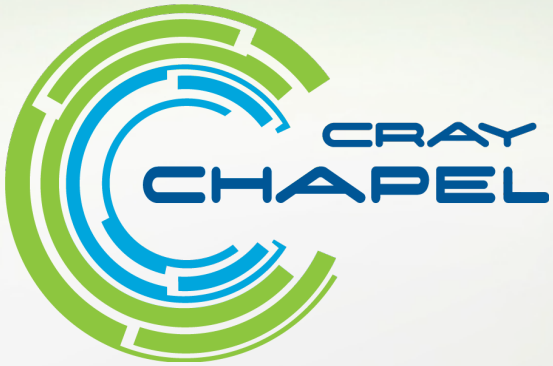
All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

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