Chapel Update

Chapel Team, Cray Inc.

SC17 Briefings

November 2017

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?

Chapel: A productive parallel programming language

- portable
- open-source
- a collaborative effort

Goals:

- Support general parallel programming
  - "any parallel algorithm on any parallel hardware"
- Make parallel programming at scale far more productive

What does “Productivity” mean to you?

Recent Graduate:

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

Seasoned HPC Programmer:

“that sugary stuff that I can’t use because I need full control to ensure good performance”

Computational Scientist:

“something that lets me express my parallel computations without requiring me to wrestle with architecture-specific details”

Chapel Team:

“something that lets the computational scientist express what they want,
without taking away the control the HPC programmer needs,
implemented in a language as attractive as recent graduates would like.”

Chapel and Other Languages

Chapel strives to be as…

…**programmable** as Python

…**fast** as Fortran

…**scalable** as MPI, SHMEM, or UPC

…**portable** as C

…**flexible** as C++

…**fun** as [your favorite programming language]

The Challenge

Q: So why don't we already have such a language already?

A: ~~Technical challenges?~~

- while they exist, we don't think this is the main issue…

A: Due to a lack, in HPC, of…

…long-term efforts

…resources

…co-design between developers and users

…community will

…patience

Chapel is our attempt to reverse this trend

A Brief History of Chapel

2002–2012: DARPA HPCS

- Cray pursued a new language, Chapel
- Delivered a compelling research prototype

2013–2018: "the 5-year push"

- Based on positive user response, Cray set out to improve Chapel
  - **performance** improvements
  - fixing / improving **features**
  - maintaining / improving **portability**
  - nurturing the **community**
  - exploring **governance** models

The Chapel Team at Cray (May 2017)

14 full-time employees + 2 summer interns + 2–4 GSoC students

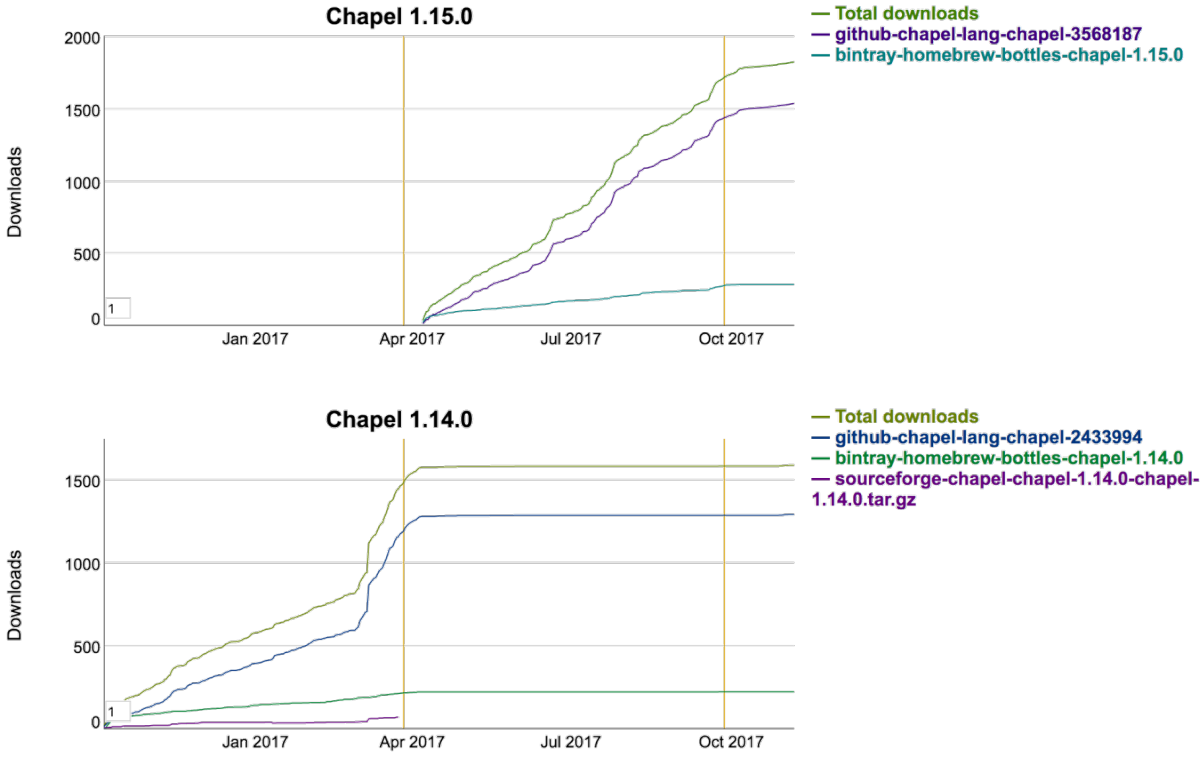
Chapel Community R&D Partners

(and several others…)

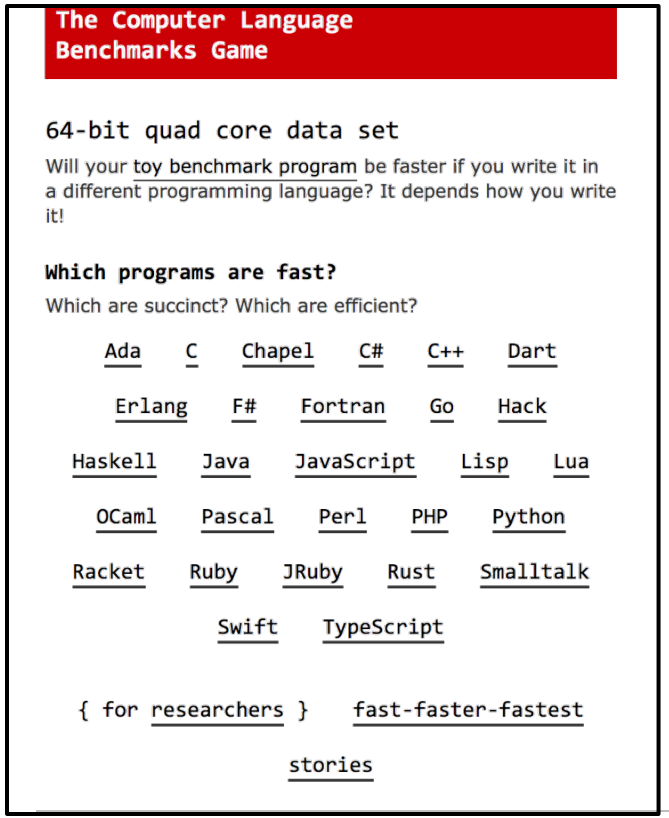
https://chapel-lang.org/collaborations.html

Highlights of the Past Year or 4½

The Year in Downloads (~3400 total, a record)



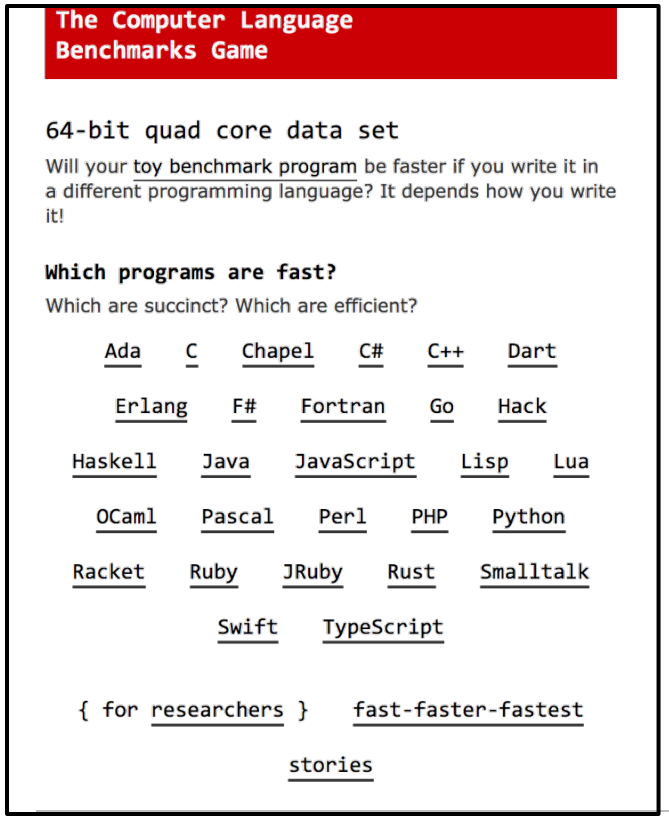
Computer Language Benchmarks Game (CLBG)



Website supporting cross-language comparisons

- 13 toy benchmark programs x
  ~28 languages x many implementations
  - exercise key computational idioms
  - specific approach prescribed

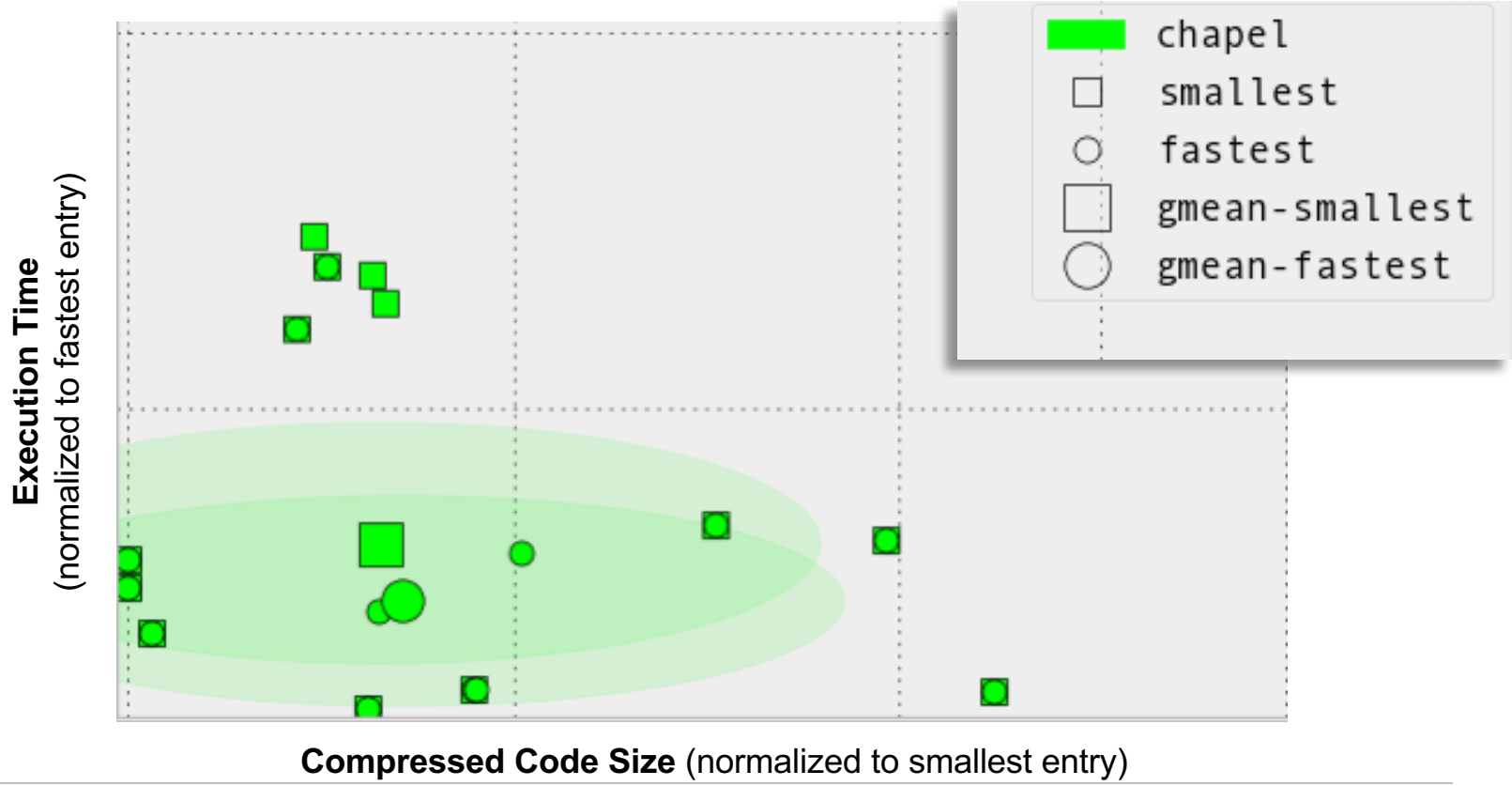
Computer Language Benchmarks Game (CLBG)



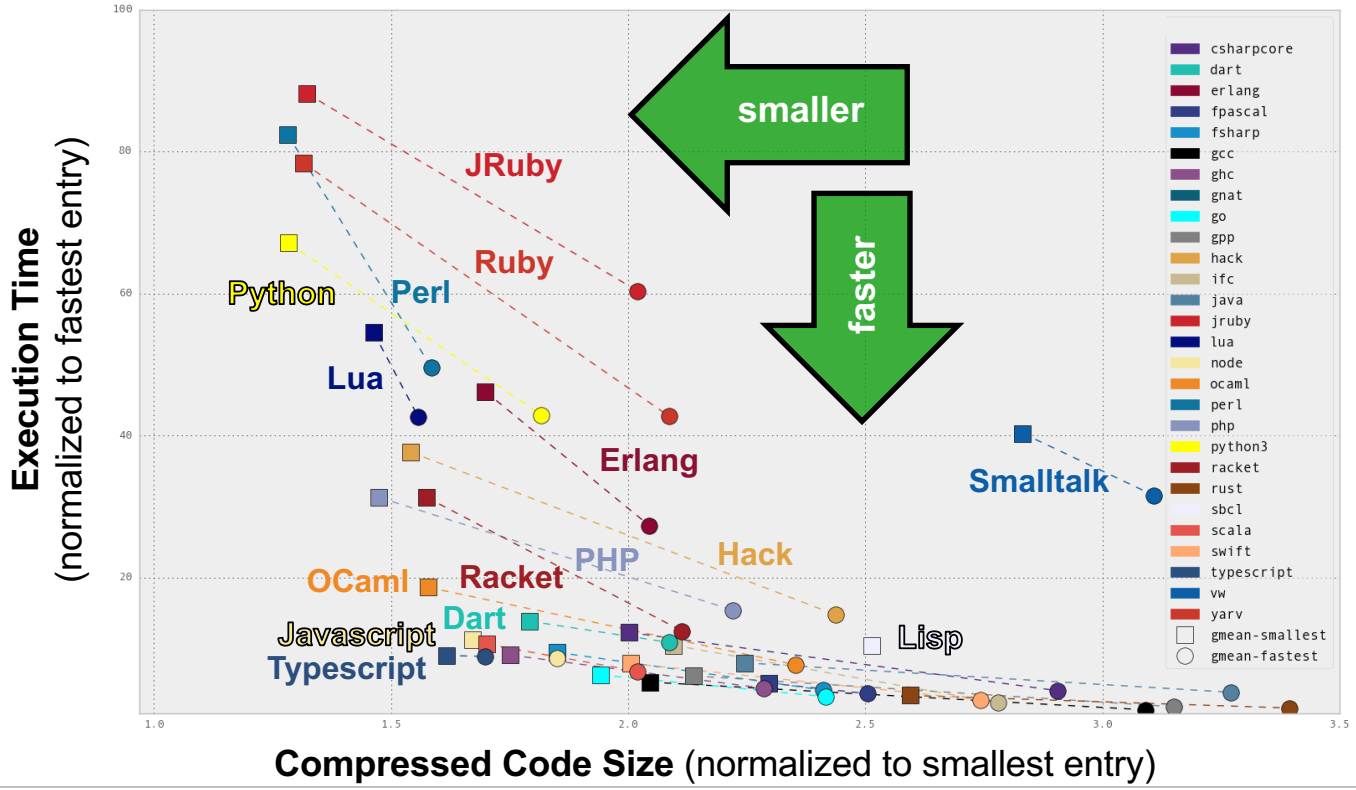
Chapel's approach to the CLBG:

- striving for elegance over heroism
  - ideally: "Want to learn how program *xyz* works? Read the Chapel version."

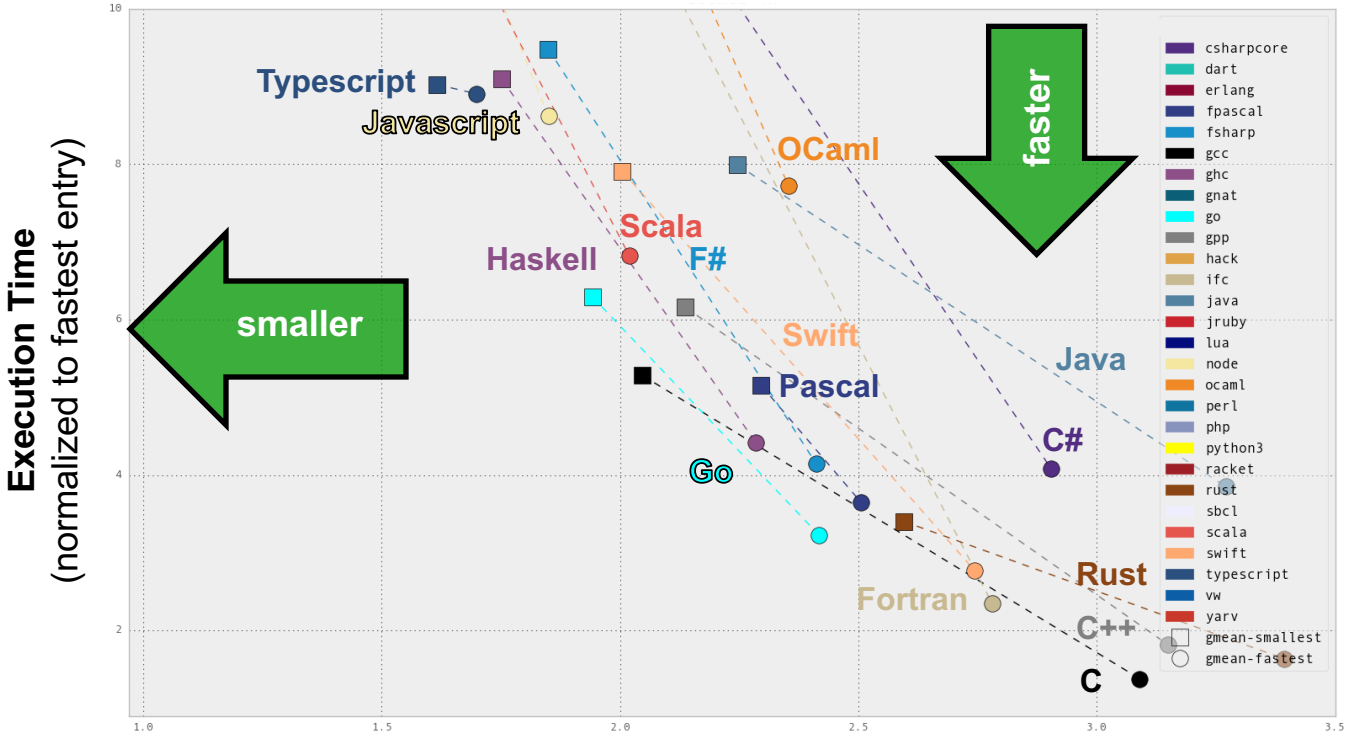
Scatter plots of CLBG code size x speed



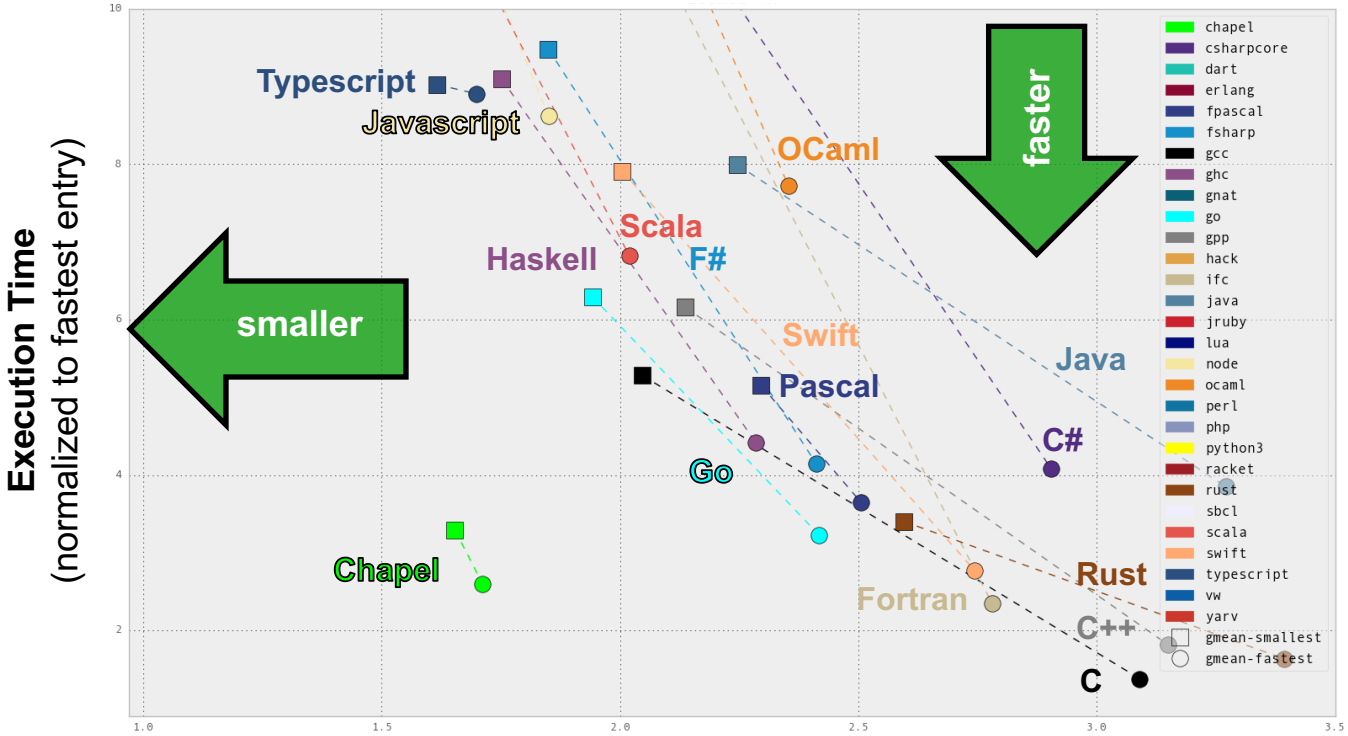
CLBG Cross-Language Summary
(Oct 2017 standings)



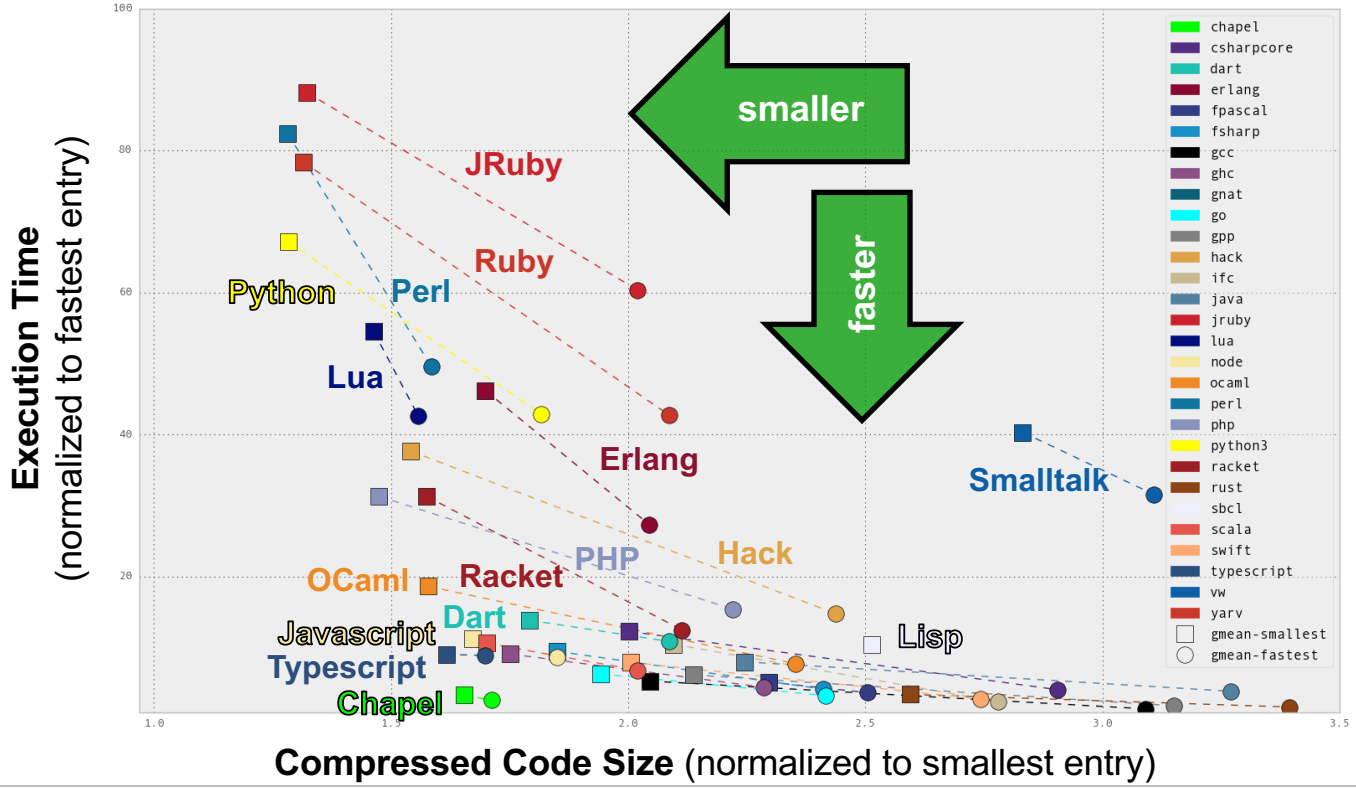
CLBG Cross-Language Summary
(Oct 2017 standings)



CLBG Cross-Language Summary
(Oct 2017 standings)



CLBG Cross-Language Summary
(Oct 2017 standings)



CLBG: Qualitative Comparisons

Can also browse program source code *(but this requires actual thought!)*:

```
proc main() {
  printColorEquations();

  const group1 = [i in 1..popSize1] new Chameneos(i, ((i-1)%3):Color);
  const group2 = [i in 1..popSize2] new Chameneos(i, colors10[i]);

  cobegin {
    holdMeetings(group1, n);
    holdMeetings(group2, n);
  }

  print(group1);
  print(group2);

  for c in group1 do delete c;
  for c in group2 do delete c;
}


//
// Print the results of getNewColor() for all color pairs.
//
proc printColorEquations() {
  for c1 in Color do
    for c2 in Color do
      writeln(c1, " + ", c2, " -> ", getNewColor(c1, c2));
  writeln();
}


//
// Hold meetings among the population by creating a shared meeting
// place, and then creating per-chameneos tasks to have meetings.
//
proc holdMeetings(population, numMeetings) {
  const place = new MeetingPlace(numMeetings);

  coforall c in population do            // create a task per chameneos
    c.haveMeetings(place, population);

  delete place;
}
```

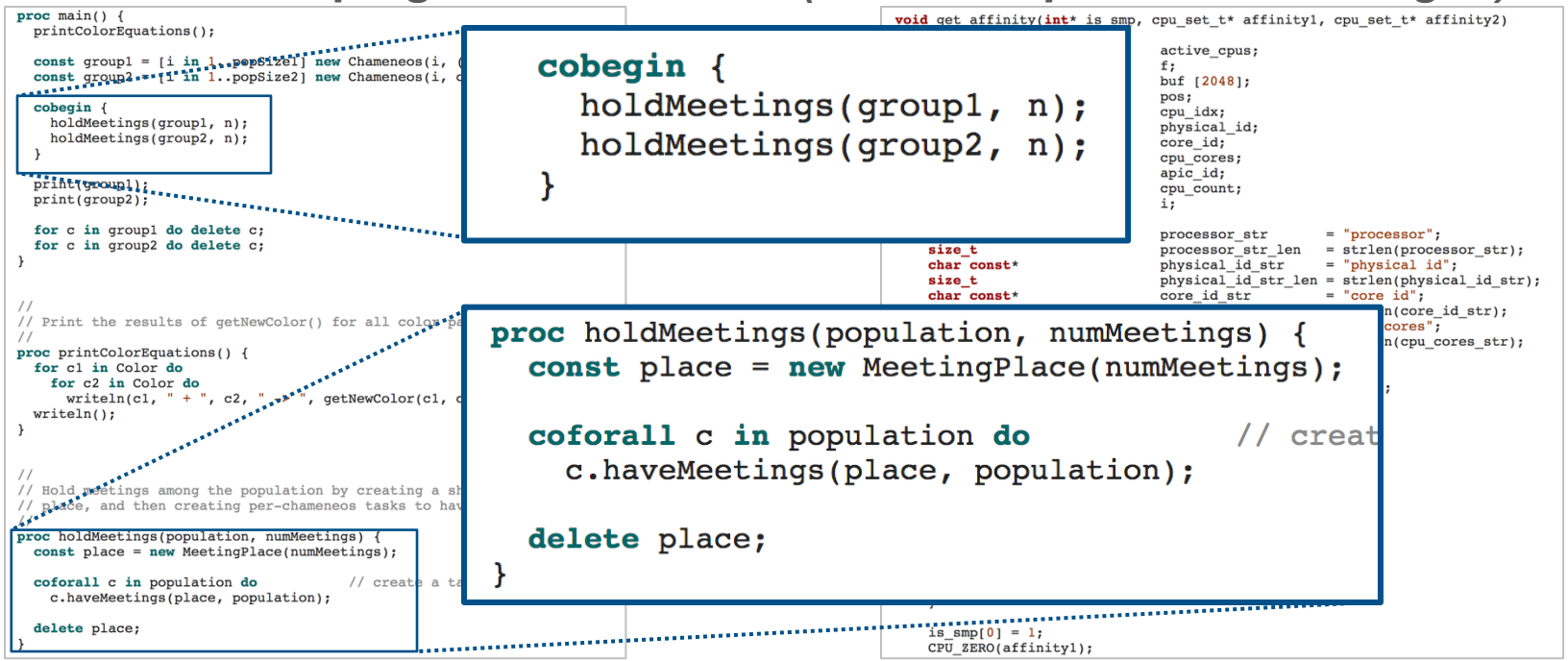
excerpt from 1210 gz Chapel entry

```
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2)
{
    cpu_set_t                   active_cpus;
    FILE*                       f;
    char                        buf [2048];
    char const*                 pos;
    int                         cpu_idx;
    int                         physical_id;
    int                         core_id;
    int                         cpu_cores;
    int                         apic_id;
    size_t                      cpu_count;
    size_t                      i;

    char const*                 processor_str       = "processor";
    size_t                      processor_str_len   = strlen(processor_str);
    char const*                 physical_id_str     = "physical id";
    size_t                      physical_id_str_len = strlen(physical_id_str);
    char const*                 core_id_str         = "core id";
    size_t                      core_id_str_len     = strlen(core_id_str);
    char const*                 cpu_cores_str       = "cpu cores";
    size_t                      cpu_cores_str_len   = strlen(cpu_cores_str);

    CPU_ZERO(&active_cpus);
    sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
    cpu_count = 0;
    for (i = 0; i != CPU_SETSIZE; i += 1)
    {
        if (CPU_ISSET(i, &active_cpus))
        {
            cpu_count += 1;
        }
    }

    if (cpu_count == 1)
    {
        is_smp[0] = 0;
        return;
    }

    is_smp[0] = 1;
    CPU_ZERO(affinity1);
```

excerpt from 2863 gz C gcc entry

CLBG: Qualitative Comparisons

Can also browse program source code *(but this requires actual thought!)*:



excerpt from 1210 gz Chapel entry

excerpt from 2863 gz C gcc entry

CLBG: Qualitative Comparisons

Can also browse program source code *(but this requires actual thought!)*:

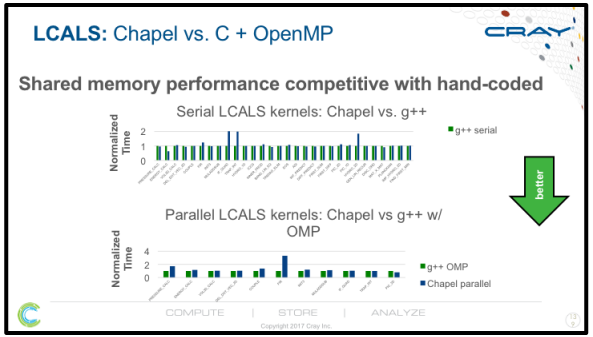
```
proc main() {
char const*                     core_id_str          = "core id"
size_t                          core_id_str_len      = strlen(co
char const*                     cpu_cores_str        = "cpu core
size_t                          cpu_cores_str_len    = strlen(cp

CPU_ZERO(&active_cpus);
sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
cpu_count = 0;
for (i = 0; i != CPU_SETSIZE; i += 1)
{
    if (CPU_ISSET(i, &active_cpus))
    {
        cpu_count += 1;
    }
}

if (cpu_count == 1)
{
    is_smp[0] = 0;
    return;
}
```

excerpt from 1210 gz Chapel entry

```
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2)
{
    cpu_set_t                   active_cpus;
    FILE*                       f;
    char                        buf [2048];
    char const*                 pos;
    int                         cpu_idx;
    int                         physical_id;
    int                         core_id;
    int                         cpu_cores;
    int                         apic_id;
    size_t                      cpu_count;
    size_t                      i;

    char const*                 processor_str       = "processor";
    size_t                      processor_str_len   = strlen(processor_str);
    char const*                 physical_id_str     = "physical id";
    size_t                      physical_id_str_len = strlen(physical_id_str);
    char const*                 core_id_str         = "core id";
    size_t                      core_id_str_len     = strlen(core_id_str);
    char const*                 cpu_cores_str       = "cpu cores";
    size_t                      cpu_cores_str_len   = strlen(cpu_cores_str);

    CPU_ZERO(&active_cpus);
    sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
    cpu_count = 0;
    for (i = 0; i != CPU_SETSIZE; i += 1)
    {
        if (CPU_ISSET(i, &active_cpus))
        {
            cpu_count += 1;
        }
    }

    if (cpu_count == 1)
    {
        is_smp[0] = 0;
        return;
    }

    is_smp[0] = 1;
    CPU_ZERO(affinity1);
```

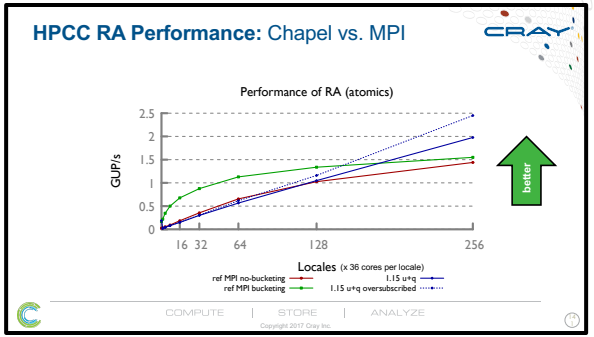
excerpt from 2863 gz C gcc entry

Chapel Performance: HPC Benchmarks



LCALS

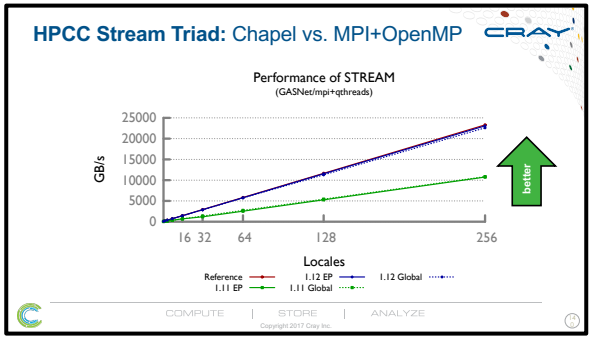
HPCC RA

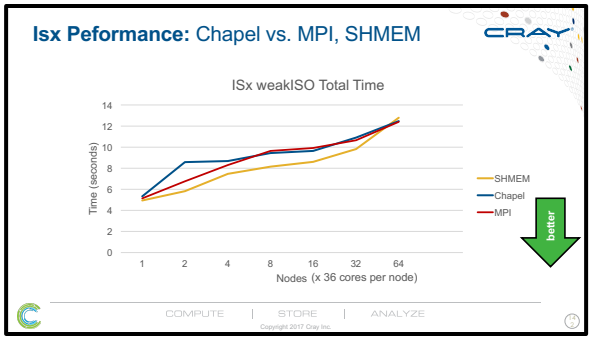


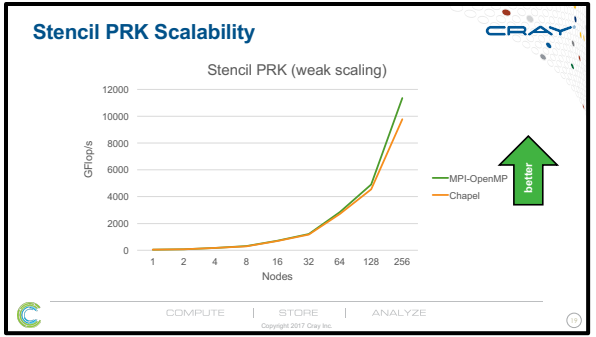
STREAM Triad

ISx

PRK Stencil



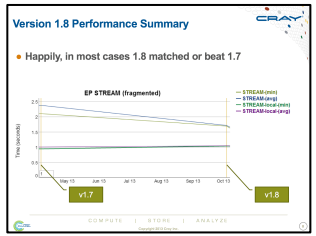


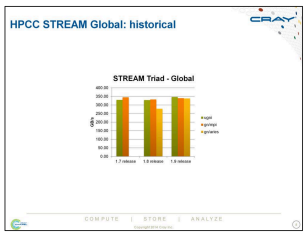


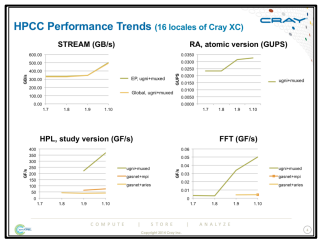
Nightly performance graphs online at: https://chapel-lang.org/perf

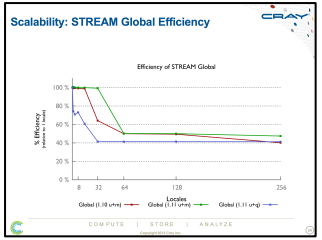
Performance: Progress Since HPCS

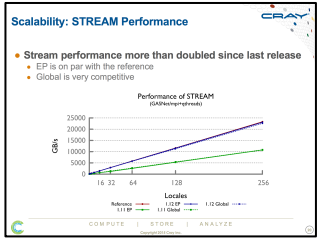
Significant improvements throughout the past 4½ years

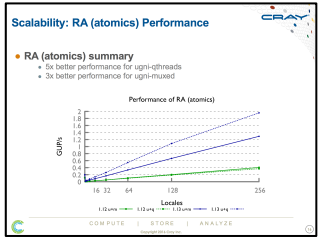


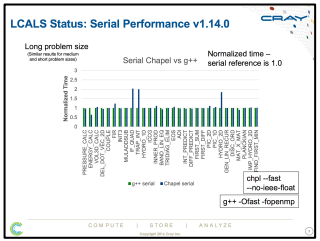


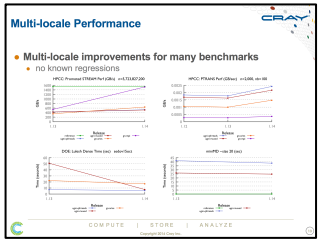


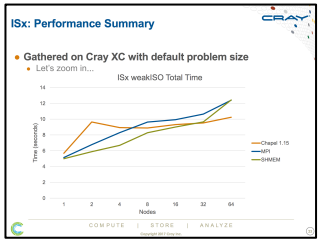


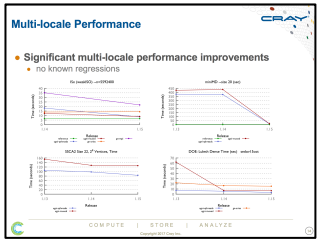


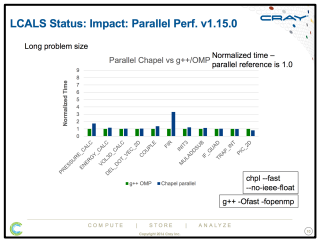


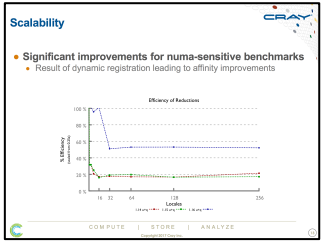












Library Highlights: Past Year

New libraries:

- Crypto
- Collections: DistributedBag, DistributedDeque
- DateTime
- DistributedIters
- Futures
- LinearAlgebra (ongoing effort)
- OwnedObject / SharedObject
- TOML (ongoing effort)

Library improvements:

- BLAS
- FFTW
- MPI
- ZMQ
- various: added ‘throw’ing versions of several routines

Library Highlights: Past Year

New libraries:

- Crypto
- Collections: DistributedBag, DistributedDeque
- DateTime
- DistributedIters
- Futures
- LinearAlgebra (ongoing effort)
- OwnedObject / SharedObject
- TOML (ongoing effort)

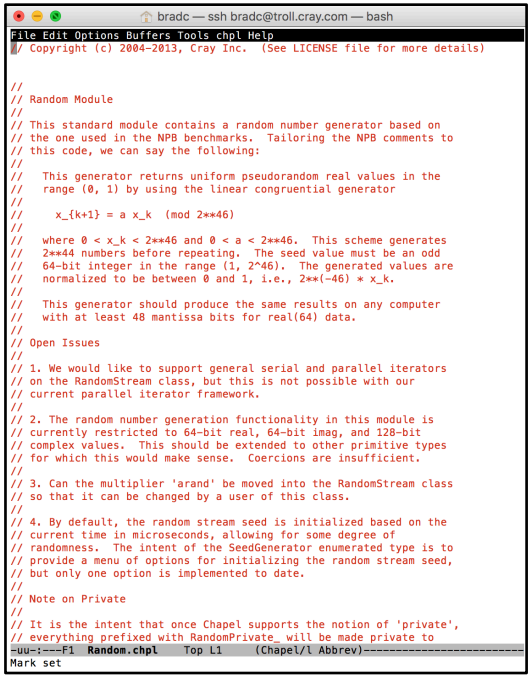
Library improvements:

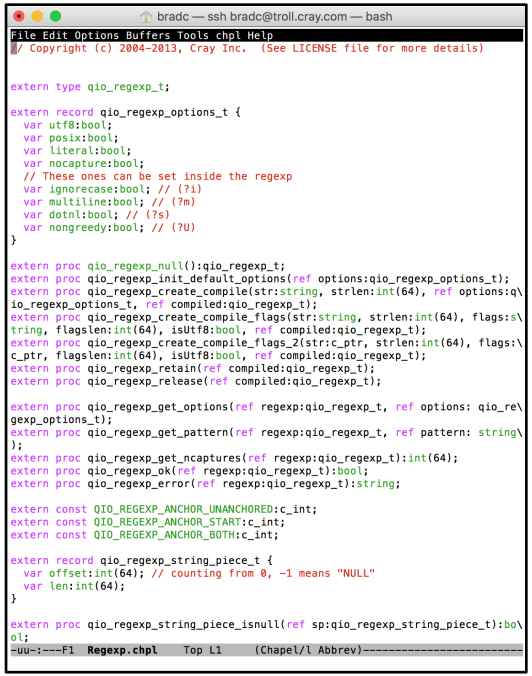
- BLAS
- FFTW
- MPI
- ZMQ
- various: added ‘throw’ing versions of several routines

(developed by GSoC student)
(developed by Cray intern)
(externally developed)

Libraries: Progress Since HPCS

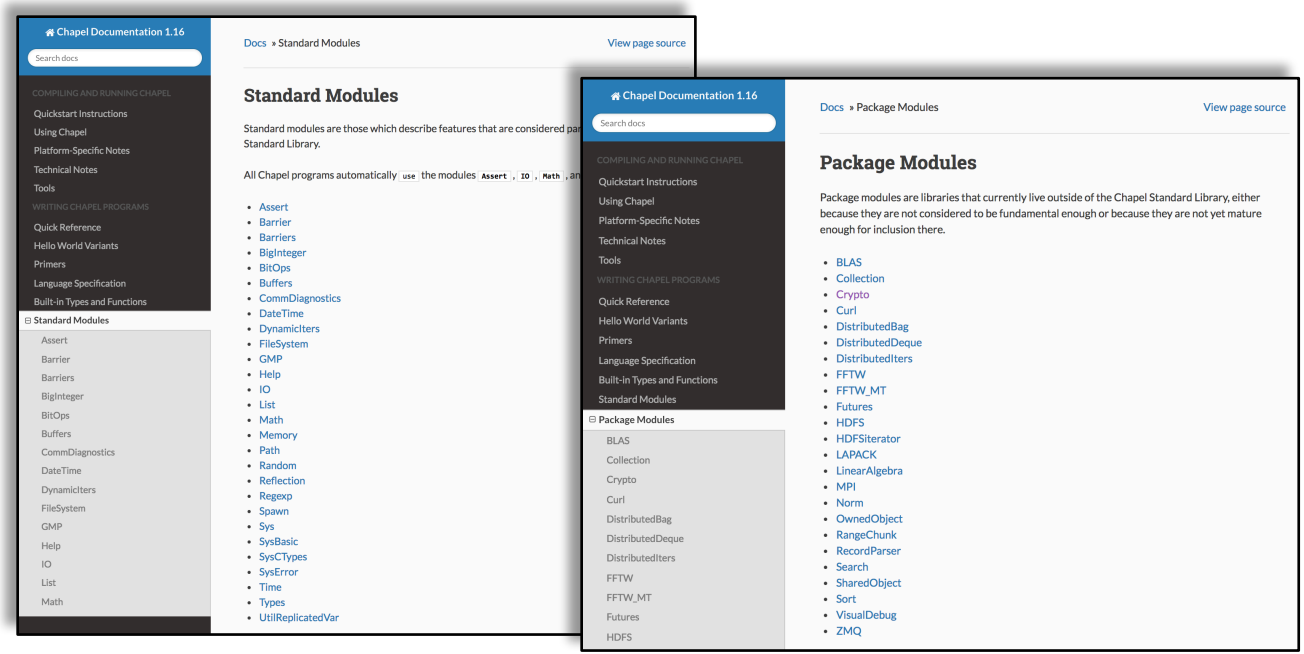
Then: ~25 modules, documented via comments (if at all)





Libraries: Progress Since HPCS

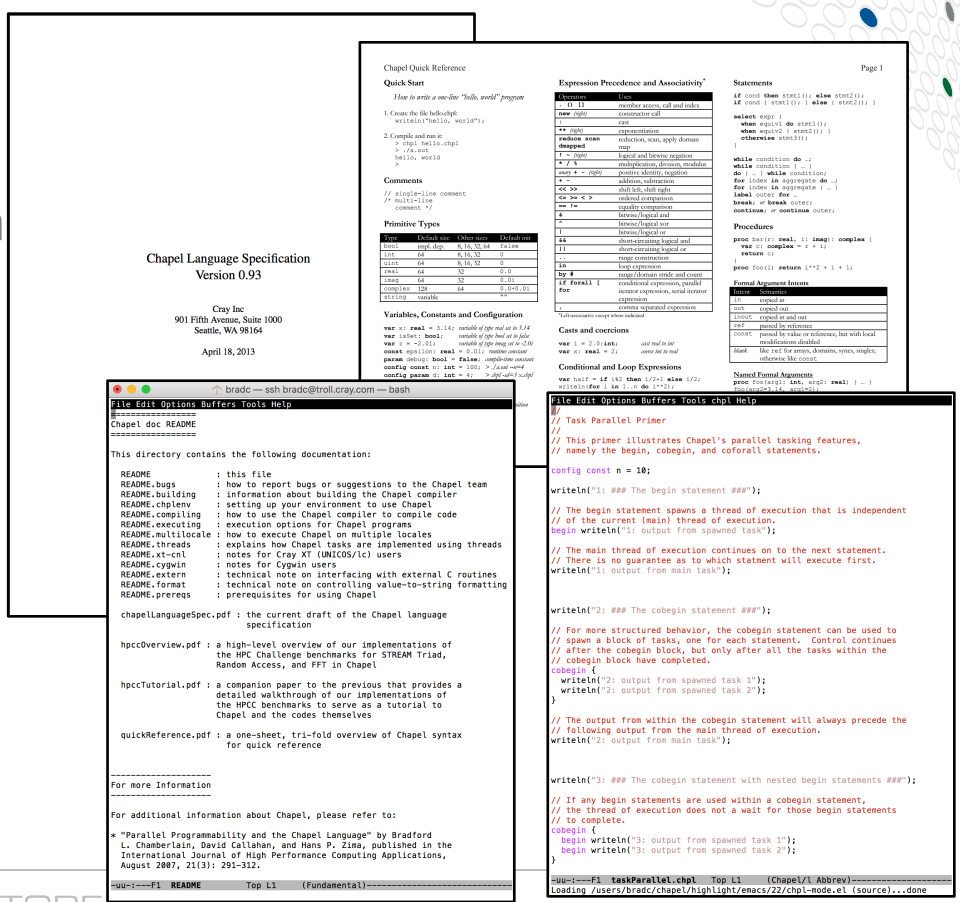
Now: ~58 documented modules, many user-contributed



Documentation: Progress Since HPCS

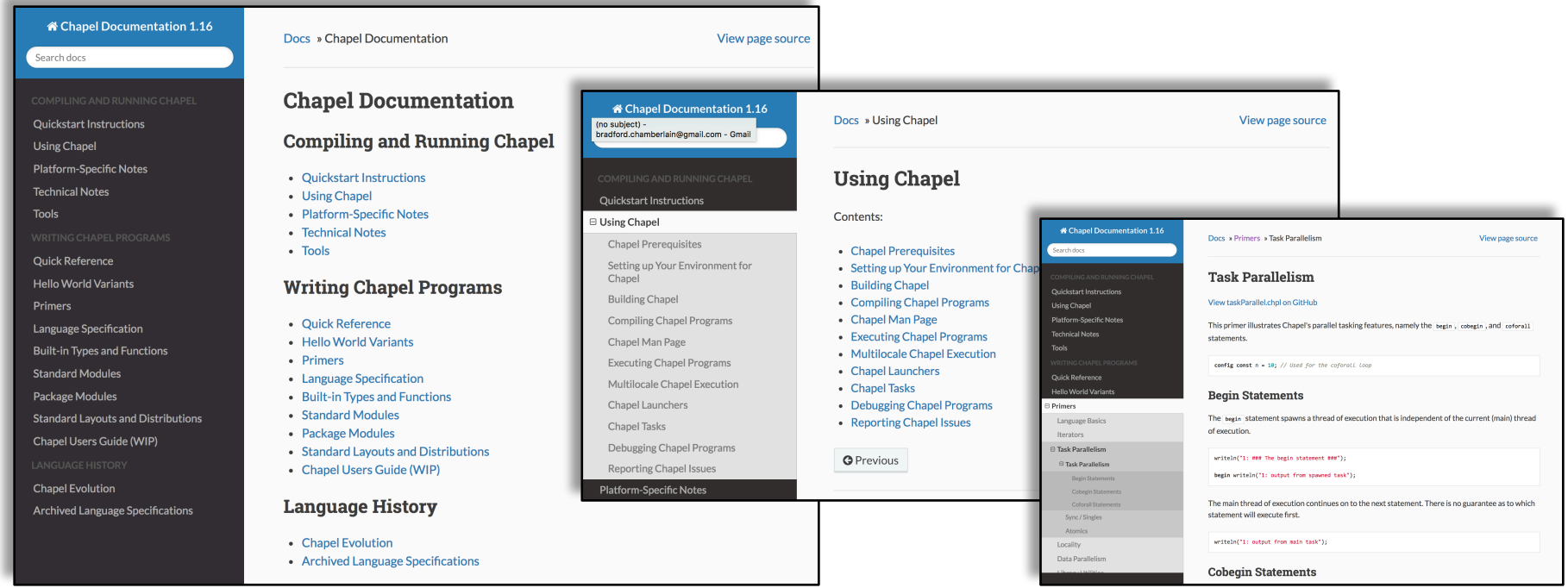
Then:

- a PDF language specification
- a Quick Reference sheet
- a number of README's
- ~22 primer examples



Documentation: Progress Since HPCS

Now: > 200 modern, hyperlinked, web-based doc pages



Tool Highlights: Past Year

- **Initial version of Chapel package manager, 'mason'**
  - modeled after Cargo, enables community to develop and share decentralized libraries

```
> mason build
Updating mason-registry
Downloading dependency: Bob-1.1.0
Downloading dependency: Alice-0.3.0
```

- **First release of 'c2chapel' tool**
  - converts C header files to Chapel 'extern' declarations

C99

```
struct allInts {
  int a;
  unsigned int b;
  long long c;
};


void msg(const char* fmt);
```

Chapel

```
extern record allInts {
  var a : c_int;
  var b : c_uint;
  var c : c_longlong;
}


extern proc msg(fmt : c_string) : void;
```

What's Next?

What's Next? (Big Ticket Items)

- **Work towards Chapel 2.0 release**
  - goal: no changes that break backwards compatibility
- **LLVM back-end by default**
- **GPU support**
- **Support for delete-free computation**
- **Application studies / application partnerships**

Crossing the Stream of Adoption

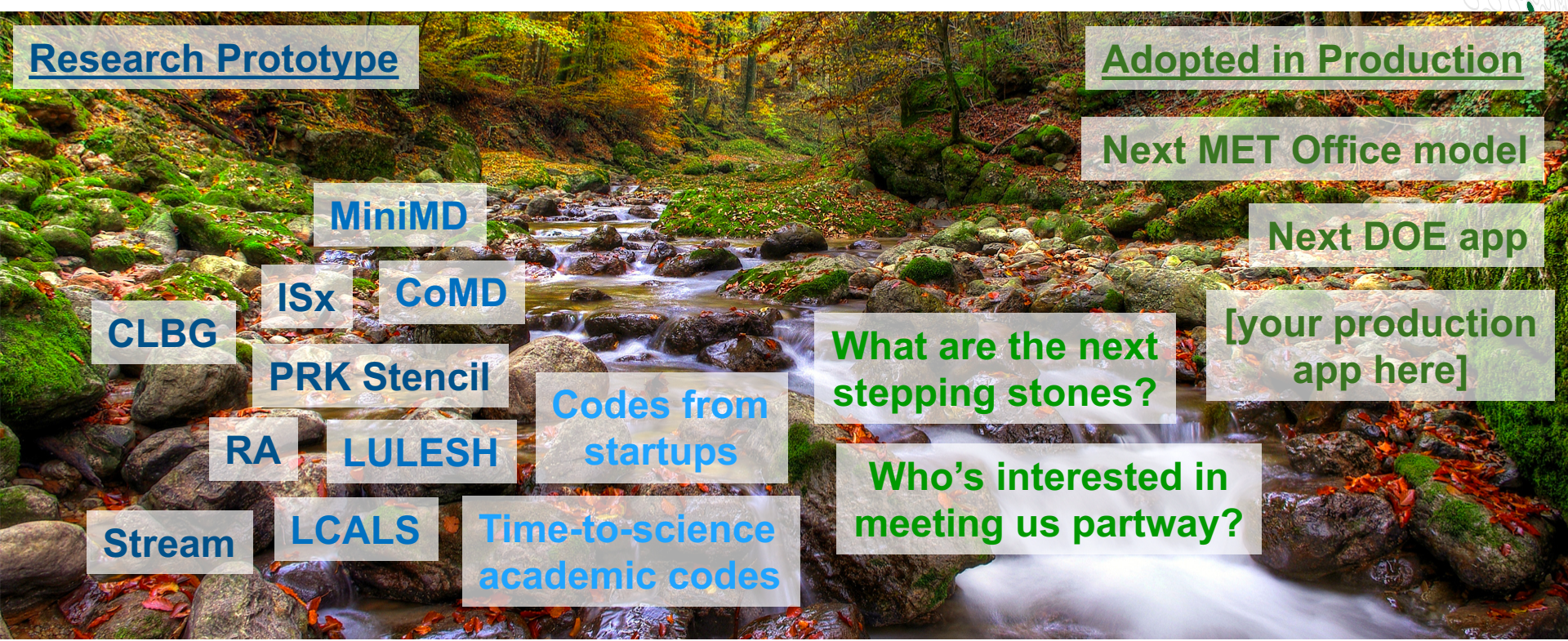
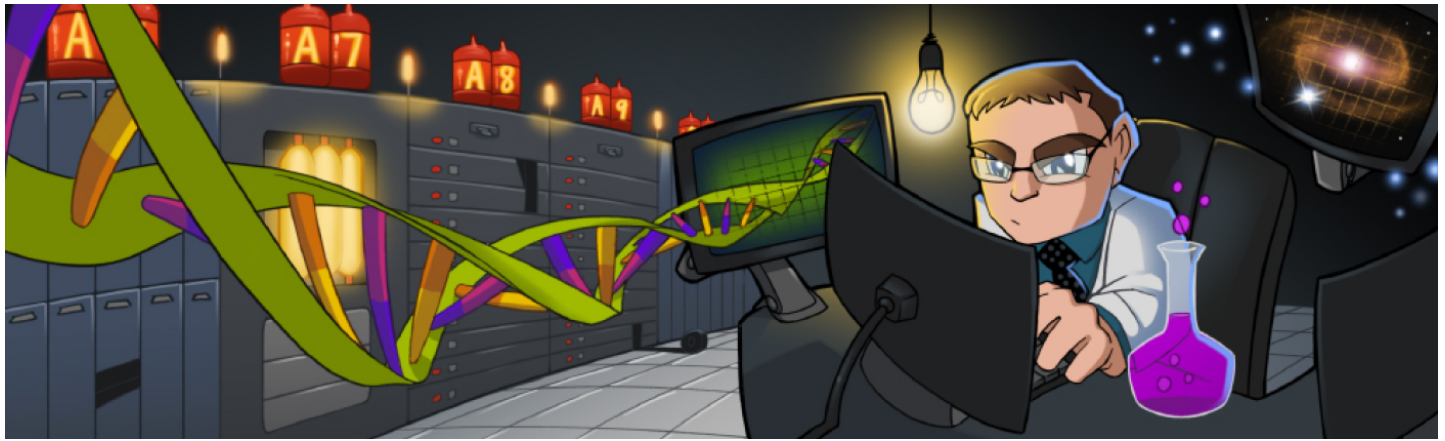


image source: http://feelgrafix.com/813578-free-stream-wallpaper.html

CHIUW 2017 Keynote

Chapel's Home in the Landscape of New Scientific Computing Languages
(and what it can learn from the neighbours)

Jonathan Dursi, *The Hospital for Sick Children, Toronto*



Quote from CHIUW 2017 keynote

"My opinion as an outsider…is that Chapel is important, Chapel is mature, and Chapel is just getting started.

"If the scientific community is going to have frameworks for solving scientific problems that are actually designed for our problems, they're going to come from a project like Chapel.

"And the thing about Chapel is that the set of all things that are 'projects like Chapel' is 'Chapel.'"

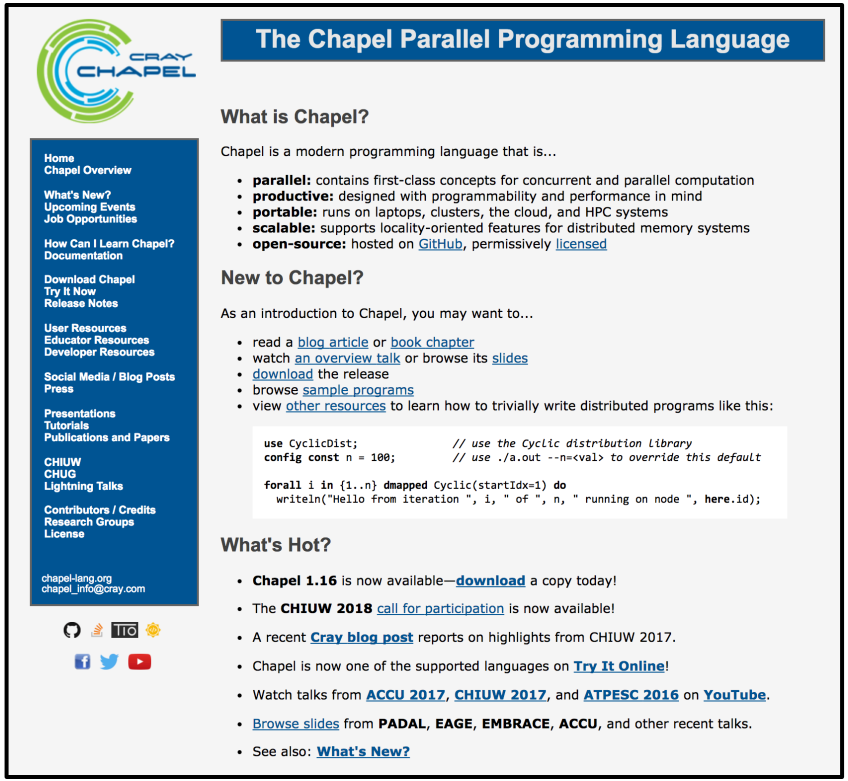
–Jonathan Dursi

Chapel's Home in the New Landscape of Scientific Frameworks
(and what it can learn from the neighbours)
CHIUW 2017 keynote

https://ljdursi.github.io/CHIUW2017 / https://www.youtube.com/watch?v=xj0rwdLOR4U

Chapel Resources

Chapel Central: https://chapel-lang.org/



The Chapel Parallel Programming Language

Home
Chapel Overview

What's New?
Upcoming Events
Job Opportunities

How Can I Learn Chapel?
Documentation

Download Chapel
Try It Now
Release Notes

User Resources
Educator Resources
Developer Resources

Social Media / Blog Posts
Press

Presentations
Tutorials
Publications and Papers

CHIUW
CHUG
Lightning Talks

Contributors / Credits
Research Groups
License

chapel-lang.org
chapel_info@cray.com

What is Chapel?

Chapel is a modern programming language that is...

- **parallel:** contains first-class concepts for concurrent and parallel computation
- **productive:** designed with programmability and performance in mind
- **portable:** runs on laptops, clusters, the cloud, and HPC systems
- **scalable:** supports locality-oriented features for distributed memory systems
- **open-source:** hosted on GitHub, permissively licensed

New to Chapel?

As an introduction to Chapel, you may want to...

- read a blog article or book chapter
- watch an overview talk or browse its slides
- download the release
- browse sample programs
- view other resources to learn how to trivially write distributed programs like this:

```
use CyclicDist;              // use the Cyclic distribution library
config const n = 100;        // use ./a.out --n=<val> to override this default

forall i in {1..n} dmapped Cyclic(startIdx=1) do
  writeln("Hello from iteration ", i, " of ", n, " running on node ", here.id);
```

What's Hot?

- **Chapel 1.16** is now available—**download** a copy today!
- The **CHIUW 2018** call for participation is now available!
- A recent **Cray blog post** reports on highlights from CHIUW 2017.
- Chapel is now one of the supported languages on **Try It Online**!
- Watch talks from **ACCU 2017**, **CHIUW 2017**, and **ATPESC 2016** on **YouTube**.
- Browse slides from **PADAL**, **EAGE**, **EMBRACE**, **ACCU**, and other recent talks.
- See also: **What's New?**

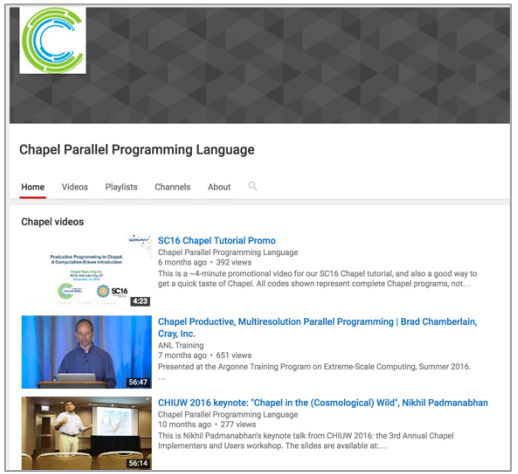
How to Stalk Chapel

http://facebook.com/ChapelLanguage

http://twitter.com/ChapelLanguage

https://www.youtube.com/channel/UCHmm27bYjhknK5mU7ZzPGsQ/

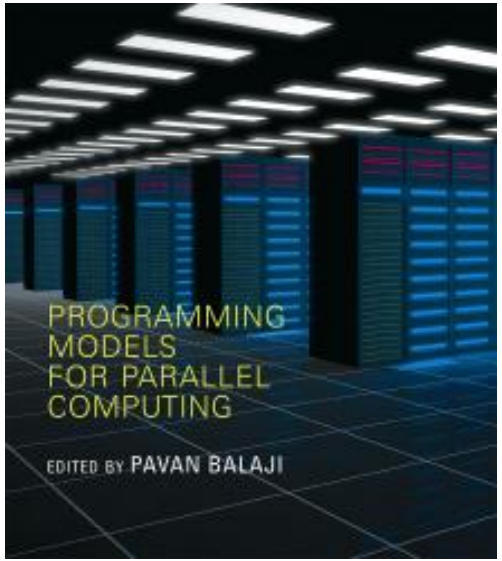
chapel-announce@lists.sourceforge.net



Suggested Reading (healthy attention spans)

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

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



Other Chapel papers/publications available at **https://chapel-lang.org/papers.html**

Suggested Reading (short attention spans)

CHIUW 2017: Surveying the Chapel Landscape, Cray Blog, July 2017.

- *a run-down of recent events*

Chapel: Productive Parallel Programming, Cray Blog, May 2013.

- *a short-and-sweet introduction to Chapel*

Six Ways to Say "Hello" in Chapel **(parts 1, 2, 3)**, Cray Blog, Sep-Oct 2015.

- *a series of articles illustrating the basics of parallelism and locality in Chapel*

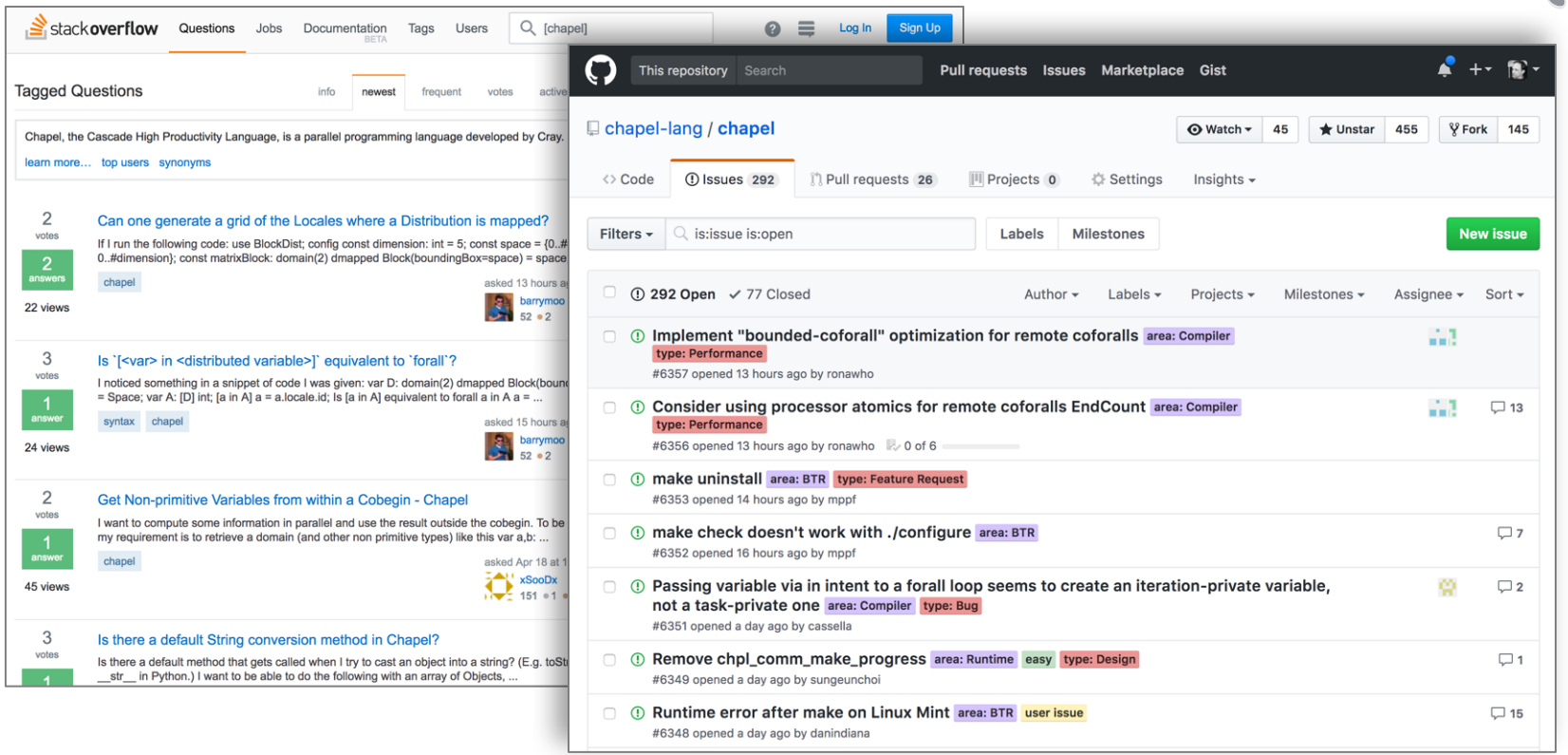
Why Chapel? **(parts 1, 2, 3)**, Cray Blog, Jun-Oct 2014.

- *a series of articles answering common questions about why we are pursuing Chapel in spite of the inherent challenges*

[Ten] Myths About Scalable Programming Languages, IEEE TCSC Blog (index available on chapel-lang.org "blog posts" page), Apr-Nov 2012.

- *a series of technical opinion pieces designed to argue against standard reasons given for not developing high-level parallel languages*

Chapel StackOverflow and GitHub Issues



Where to..

Submit bug reports:

GitHub issues for chapel-lang/chapel: public bug forum
chapel_bugs@cray.com: for reporting non-public bugs

Ask User-Oriented Questions:

StackOverflow: when appropriate / other users might care
#chapel-users (irc.freenode.net): user-oriented IRC channel
chapel-users@lists.sourceforge.net: user discussions

Discuss Chapel development

chapel-developers@lists.sourceforge.net: developer discussions
#chapel-developers (irc.freenode.net): developer-oriented IRC channel

Discuss Chapel's use in education

chapel-education@lists.sourceforge.net: educator discussions

Directly contact Chapel team at Cray: chapel_info@cray.com

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