Chapel

Open-Source Productive Parallel Programming at Scale

Ben Albrecht and Brad Chamberlain

Chapel Team, Cray Inc.

OpenSuCo 2017, SC17, November 12, 2017

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

Chapel and Productivity

Chapel strives to be…

…as programmable as Python

…as fast as Fortran

…as scalable as MPI (or SHMEM or UPC)

…as portable as C

…as flexible as C++

…as fun as [your favorite programming language]

The Chapel Team at Cray (May 2017)

14 full-time employees + 2 summer interns

Chapel Community R&D Partners

HAVERFORD COLLEGE

AMD

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

WESTERN WASHINGTON UNIVERSITY

東京大学 THE UNIVERSITY OF TOKYO

THE UNIVERSITY OF ARIZONA

RICE

UNIVERSITY OF MARYLAND

BERKELEY LAB
Lawrence Berkeley National Laboratory

Lawrence Livermore National Laboratory

Sandia National Laboratories

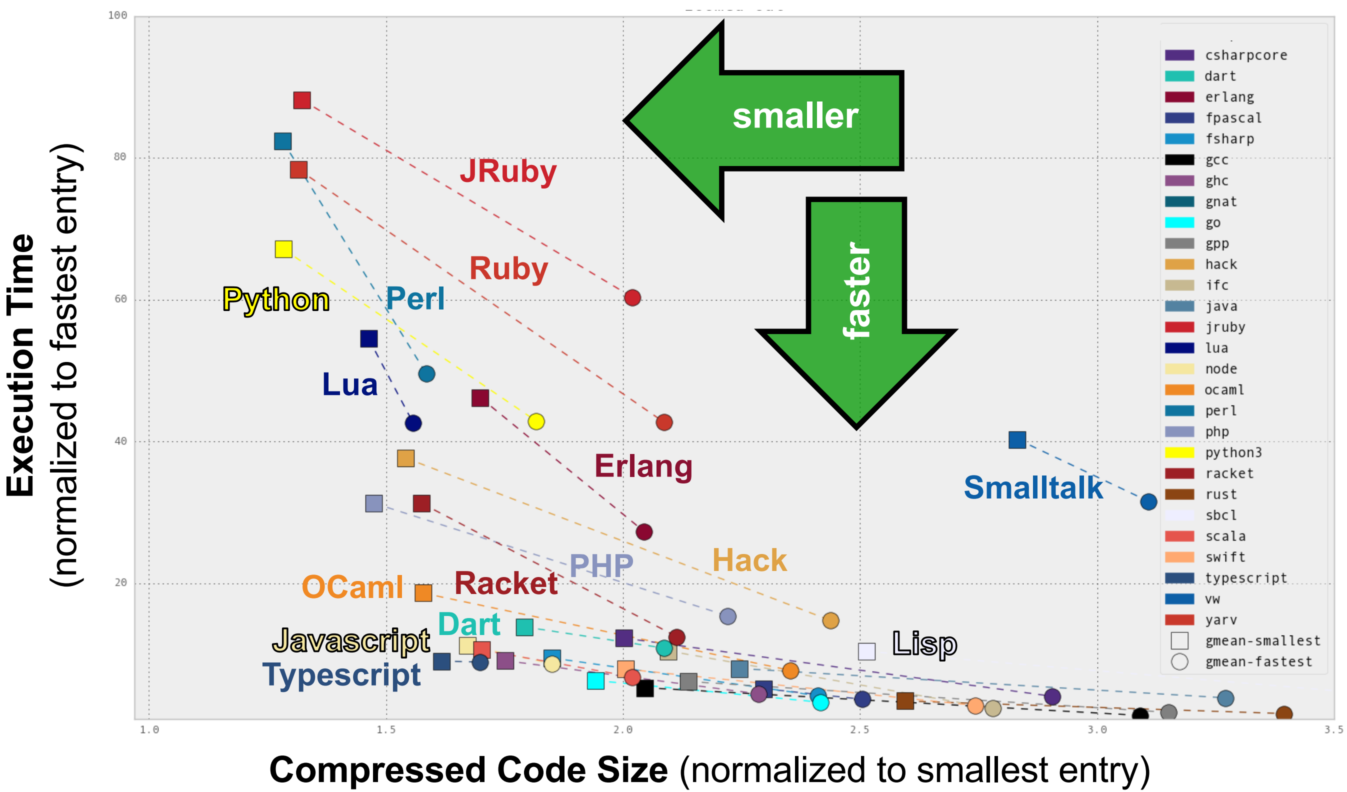
DEPARTMENT OF DEFENSE · UNITED STATES OF AMERICA

Yale

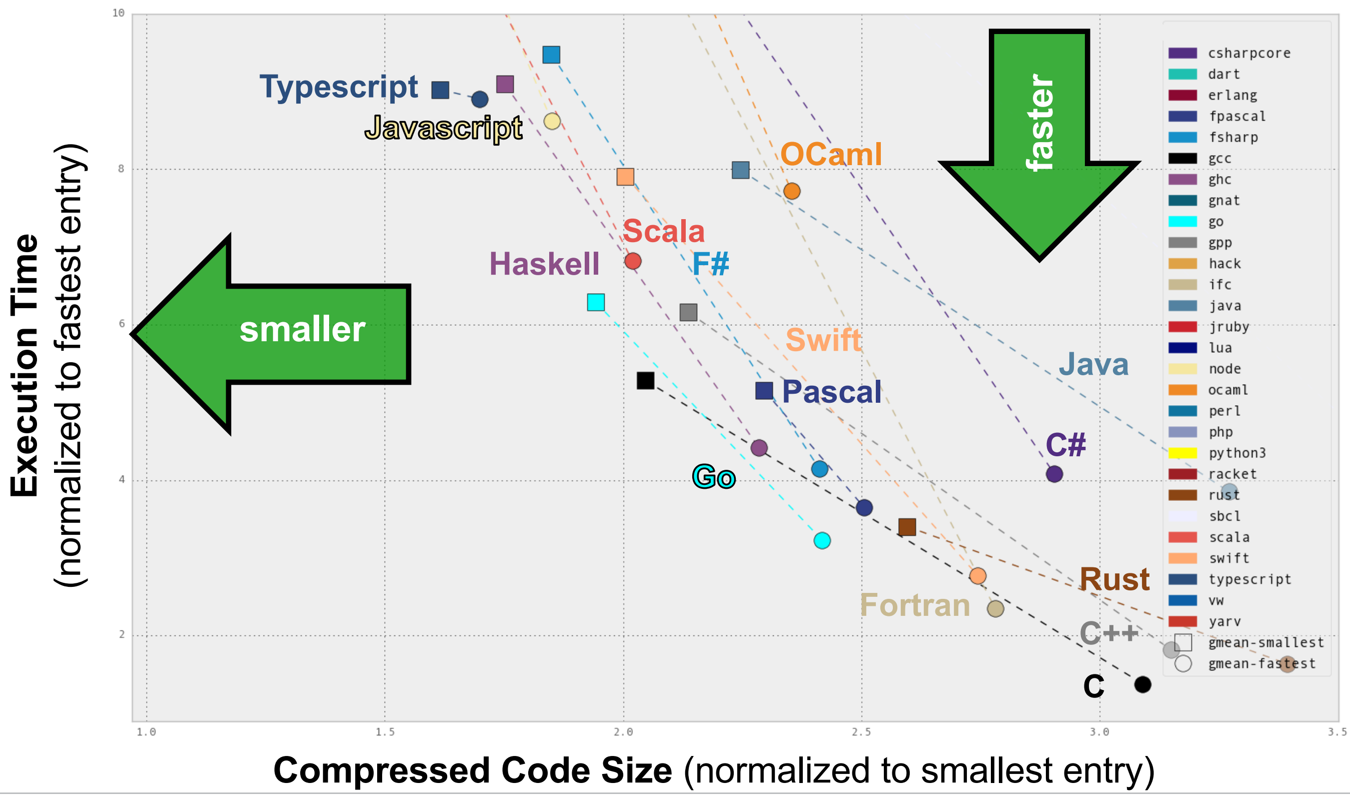
(and several others…)

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

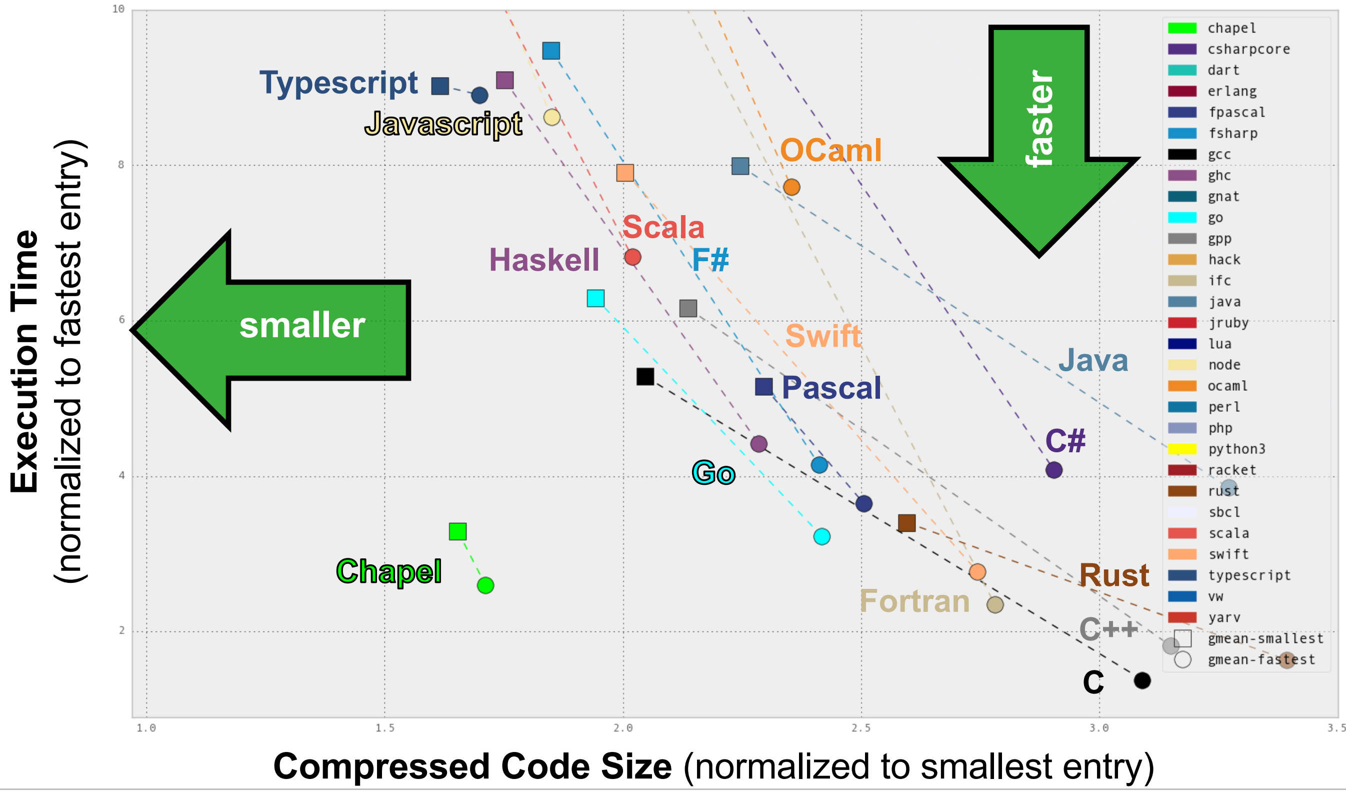
CLBG Language Cross-Language Summary
(Oct 2017 standings)



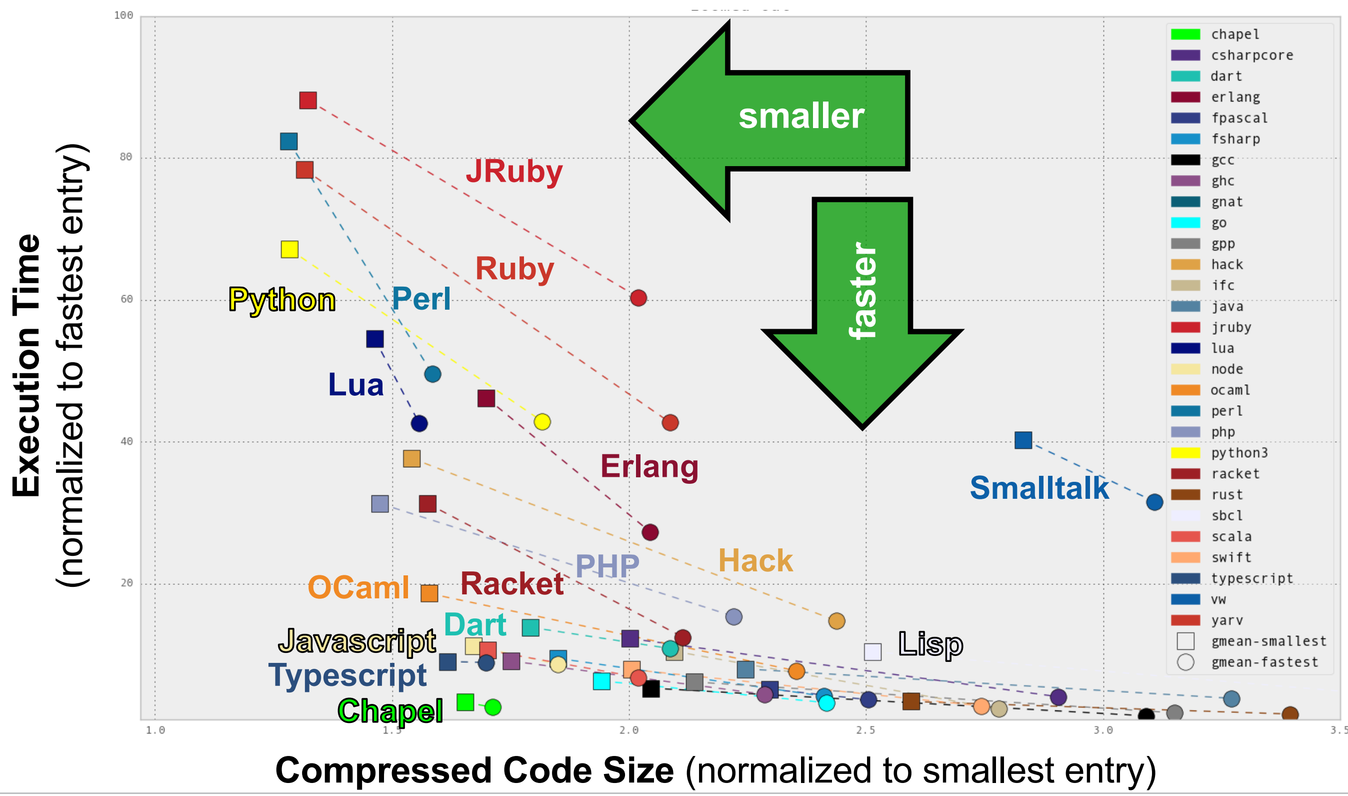
CLBG Language Cross-Language Summary
(Oct 2017 standings, zoomed in)



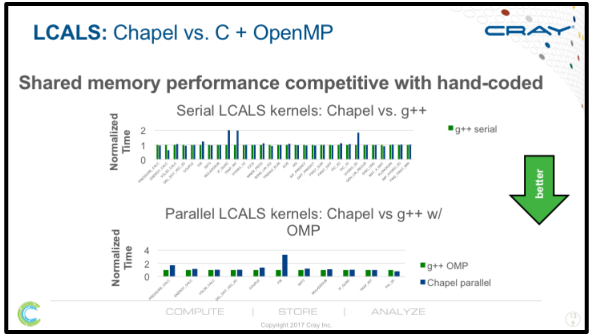
CLBG Language Cross-Language Summary
(Oct 2017 standings, zoomed in)



CLBG Language Cross-Language Summary
(Oct 2017 standings)



Chapel Performance: Competitive for HPC

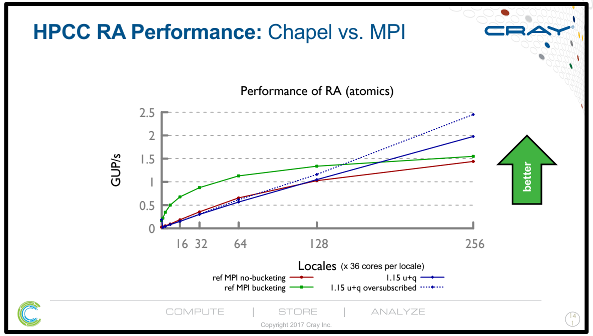


LCALS: Chapel vs. C + OpenMP

Shared memory performance competitive with hand-coded

LCALS

HPCC RA

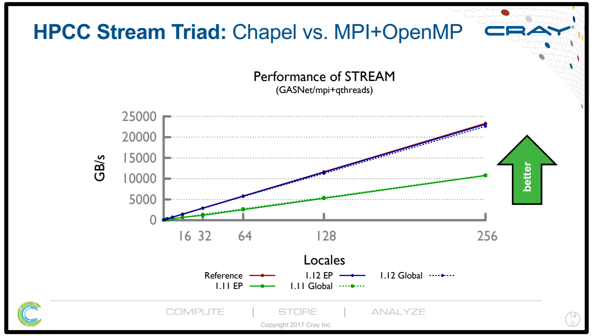


HPCC RA Performance: Chapel vs. MPI

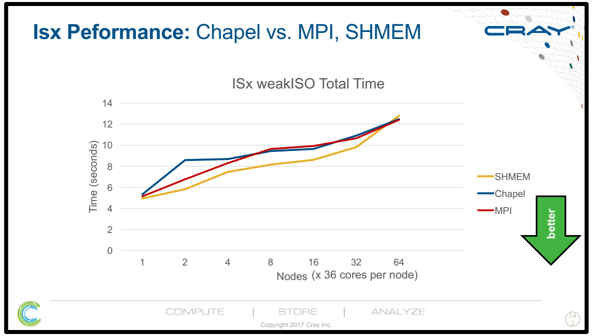
STREAM Triad

ISx

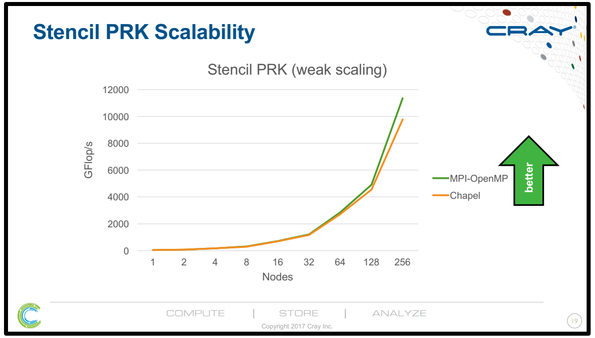
PRK Stencil



HPCC Stream Triad: Chapel vs. MPI+OpenMP



Isx Peformance: Chapel vs. MPI, SHMEM



Stencil PRK Scalability

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

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

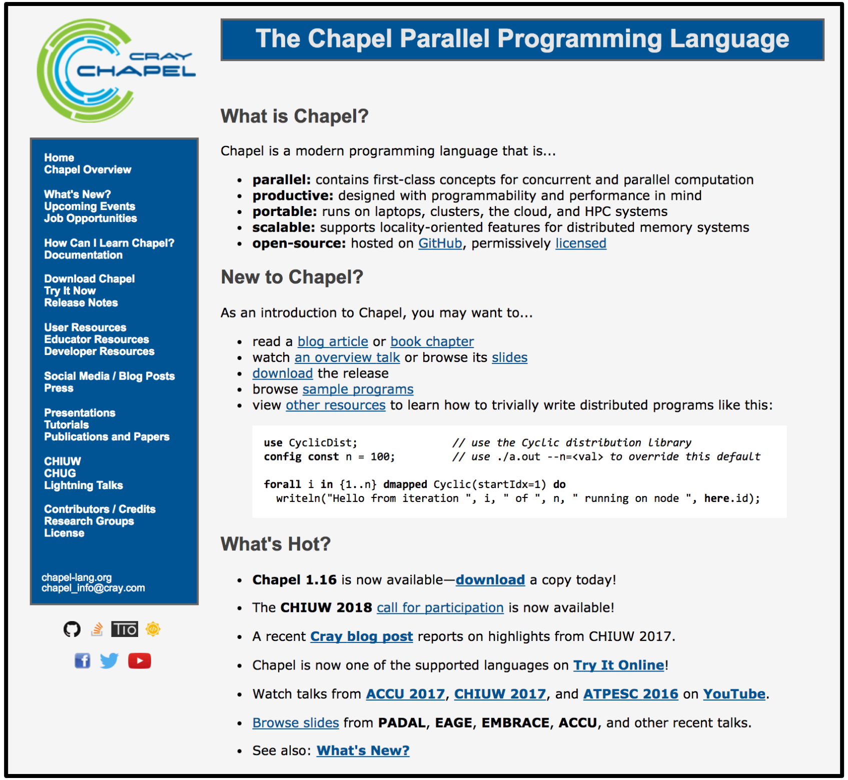
Demo Time

pre-recorded terminal sessions available online:

- installing via homebrew: https://asciinema.org/a/147072
- basics and task parallelism: https://asciinema.org/a/147073
- locality and task parallelism: https://asciinema.org/a/147135
- data parallelism: https://asciinema.org/a/147082

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?**

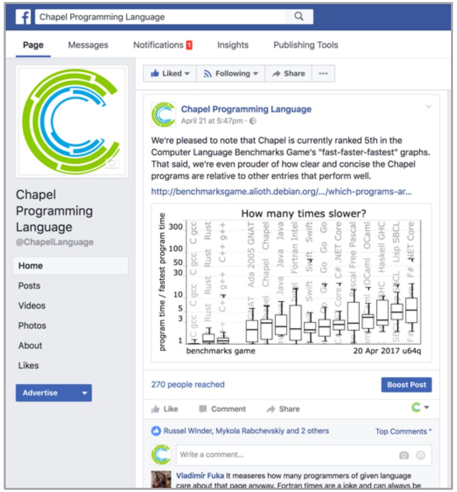
How to Stalk Chapel

http://facebook.com/ChapelLanguage

http://twitter.com/ChapelLanguage

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

chapel-announce@lists.sourceforge.net



Chapel Programming Language
@ChapelLanguage

Chapel Programming Language
April 21 at 5:47pm

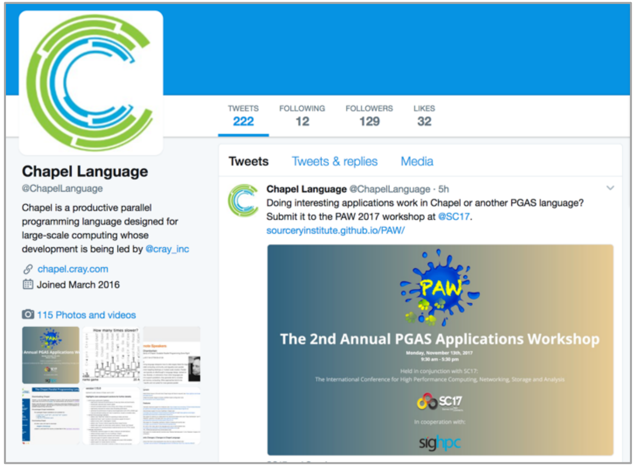
We're pleased to note that Chapel is currently ranked 5th in the Computer Language Benchmarks Game's "fast-faster-fastest" graphs. That said, we're even prouder of how clear and concise the Chapel programs are relative to other entries that perform well.

http://benchmarksgame.alioth.debian.org/.../which-programs-ar...

270 people reached

Russel Winder, Mykola Rabchevskiy and 2 others

Vladimír Fuka It measeres how many programmers of given language care about that page anyway. Fortran times are a joke and can always be


TWEETS
222
FOLLOWING
12
FOLLOWERS
129
LIKES
32
Tweets
Tweets & replies
Media
PAW
The 2nd Annual PGAS Applications Workshop
Monday, November 13th, 2017
9:30 am - 5:30 pm
Held in conjunction with SC17:
The International Conference for High Performance Computing, Networking, Storage and Analysis
SC17
In cooperation with:
sighpc

Chapel Language
@ChapelLanguage

Chapel is a productive parallel programming language designed for large-scale computing whose development is being led by @cray_inc

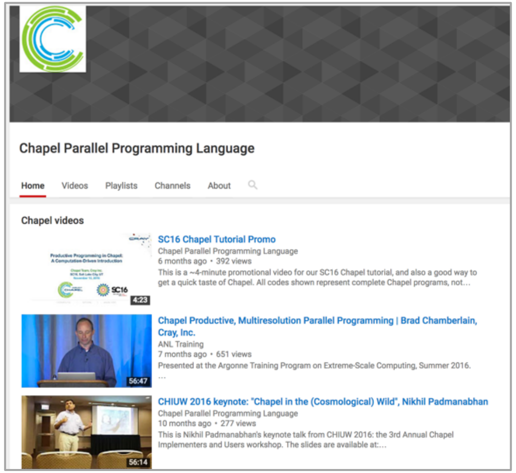
chapel.cray.com

Joined March 2016

115 Photos and videos

Chapel Language @ChapelLanguage · 5h

Doing interesting applications work in Chapel or another PGAS language? Submit it to the PAW 2017 workshop at @SC17.
sourceryinstitute.github.io/PAW/



Chapel Parallel Programming Language

Chapel videos

SC16 Chapel Tutorial Promo
Chapel Parallel Programming Language
6 months ago • 392 views
This is a ~4-minute promotional video for our SC16 Chapel tutorial, and also a good way to get a quick taste of Chapel. All codes shown represent complete Chapel programs, not...

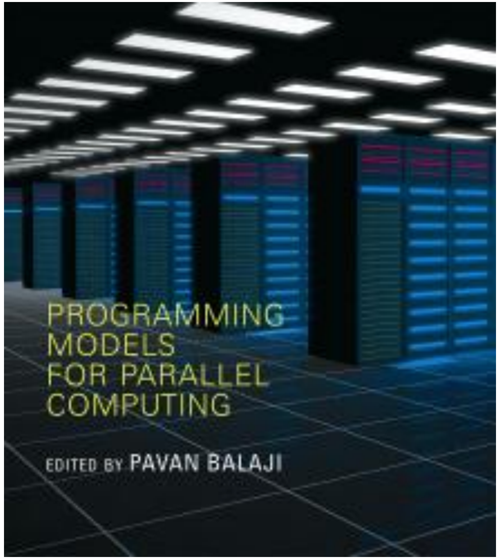
Chapel Productive, Multiresolution Parallel Programming | Brad Chamberlain, Cray, Inc.
ANL Training
7 months ago • 651 views
Presented at the Argonne Training Program on Extreme-Scale Computing, Summer 2016.
...

CHIUW 2016 keynote: "Chapel in the (Cosmological) Wild", Nikhil Padmanabhan
Chapel Parallel Programming Language
10 months ago • 277 views
This is Nikhil Padmanabhan's keynote talk from CHIUW 2016: the 3rd Annual Chapel Implementers and Users workshop. The slides are available at:...

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*

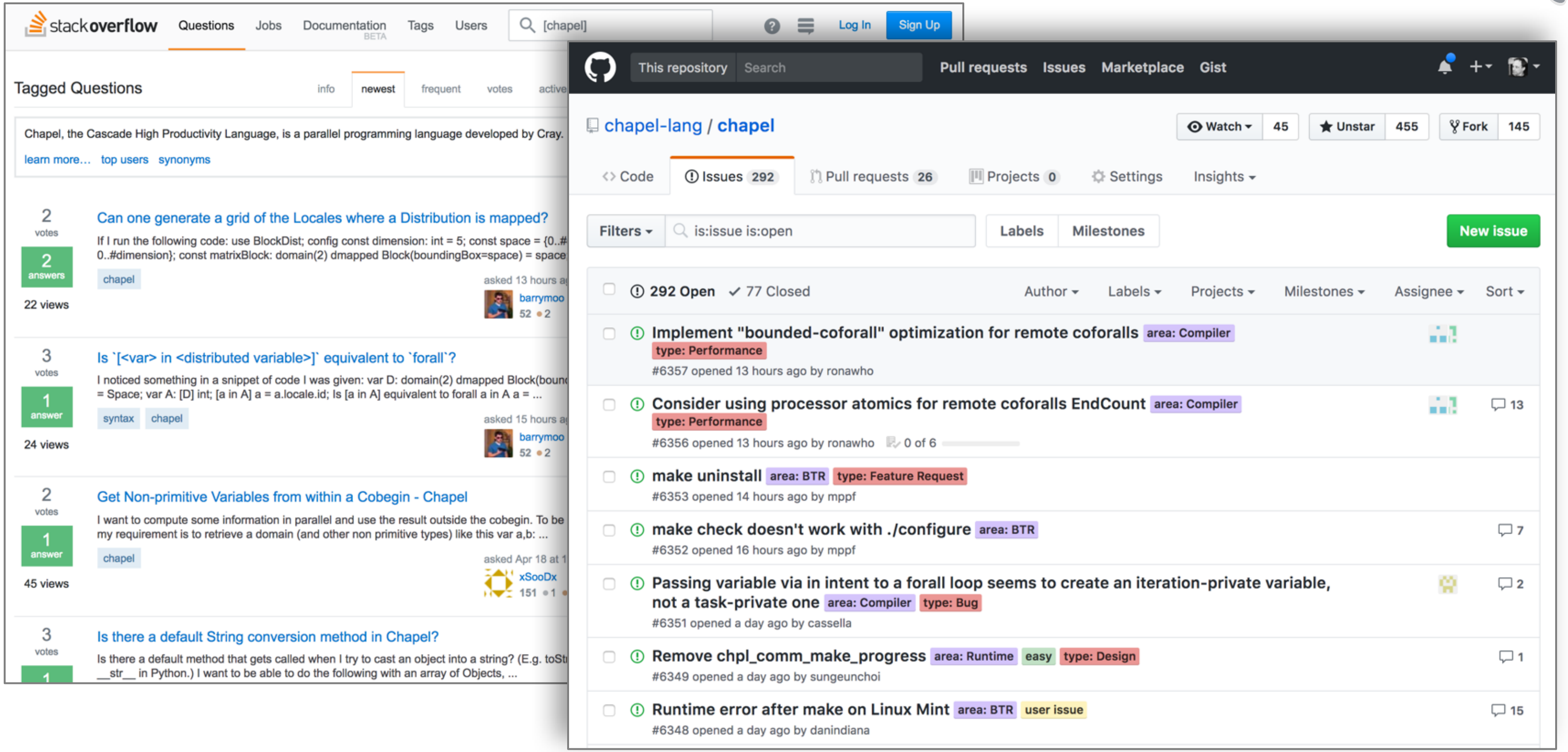
Why Chapel? **(parts 1, 2, 3)**, Cray Blog, Jun-Oct 2014.

- *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



Tagged Questions

Chapel, the Cascade High Productivity Language, is a parallel programming language developed by Cray.

learn more… top users synonyms

2 votes, 2 answers, 22 views
Can one generate a grid of the Locales where a Distribution is mapped?
If I run the following code: use BlockDist; config const dimension: int = 5; const space = {0..#
0..#dimension}; const matrixBlock: domain(2) dmapped Block(boundingBox=space) = space
chapel
asked 13 hours a
barrymoo 52 ●2

3 votes, 1 answer, 24 views
Is `[<var> in <distributed variable>]` equivalent to `forall`?
I noticed something in a snippet of code I was given: var D: domain(2) dmapped Block(bound
= Space; var A: [D] int; [a in A] a = a.locale.id; Is [a in A] equivalent to forall a in A a = ...
syntax chapel
asked 15 hours a
barrymoo 52 ●2

2 votes, 1 answer, 45 views
Get Non-primitive Variables from within a Cobegin - Chapel
I want to compute some information in parallel and use the result outside the cobegin. To be
my requirement is to retrieve a domain (and other non primitive types) like this var a,b: ...
chapel
asked Apr 18 at 1
xSooDx 151 ●1

3 votes, 1
Is there a default String conversion method in Chapel?
Is there a default method that gets called when I try to cast an object into a string? (E.g. toSt
__str__ in Python.) I want to be able to do the following with an array of Objects, ...

chapel-lang / chapel

□ ⓘ 292 Open ✓ 77 Closed

- **Implement "bounded-coforall" optimization for remote coforalls** area: Compiler type: Performance
  #6357 opened 13 hours ago by ronawho
- **Consider using processor atomics for remote coforalls EndCount** area: Compiler type: Performance — 13
  #6356 opened 13 hours ago by ronawho 0 of 6
- **make uninstall** area: BTR type: Feature Request
  #6353 opened 14 hours ago by mppf
- **make check doesn't work with ./configure** area: BTR — 7
  #6352 opened 16 hours ago by mppf
- **Passing variable via in intent to a forall loop seems to create an iteration-private variable, not a task-private one** area: Compiler type: Bug — 2
  #6351 opened a day ago by cassella
- **Remove chpl_comm_make_progress** area: Runtime easy type: Design — 1
  #6349 opened a day ago by sungeunchoi
- **Runtime error after make on Linux Mint** area: BTR user issue — 15
  #6348 opened a day ago by danindiana

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 CHAPEL

CRAY

THE SUPERCOMPUTER COMPANY