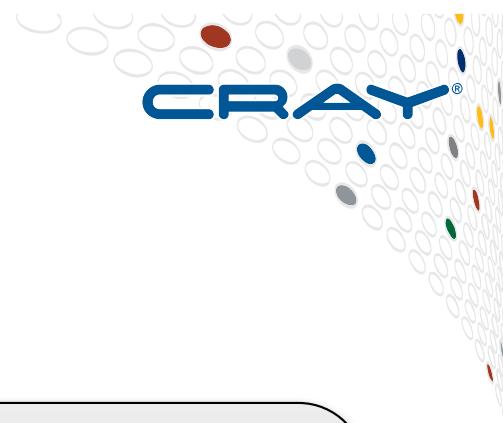


CHIUW 2017: Welcome and State of the Project

Brad Chamberlain
Chapel Team, Cray Inc.
June 2, 2017





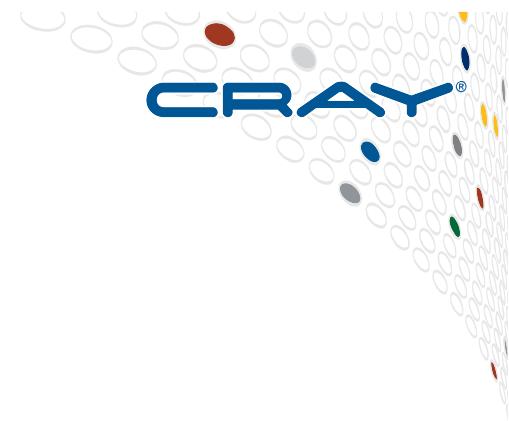
Safe Harbor Statement

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.



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Chapel, briefly



COMPUTE

|

STORE

|

ANALYZE

Copyright 2017 Cray Inc.



What is Chapel?

Chapel: A productive parallel programming language

- portable
- open-source
- a collaborative effort

Goals:

- Support general parallel programming
- Make parallel programming at scale far more productive



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Motivation for Chapel

Q: Can a single language be...

- ...as programmable as Python?
- ...as fast as Fortran?
- ...as portable as C?
- ...as scalable as MPI?
- ...as generic and meta- as C++? (but using simpler notation?)
- ...as fun as <your favorite language here>?

A: We believe so.

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

A: Due to a lack of...

- ...long-term efforts
- ...resources
- ...community will
- ...developer/user co-design
- ...patience

Chapel is our attempt
to change this



COMPUTE

|

STORE

|

ANALYZE

A Year in the Life of Chapel

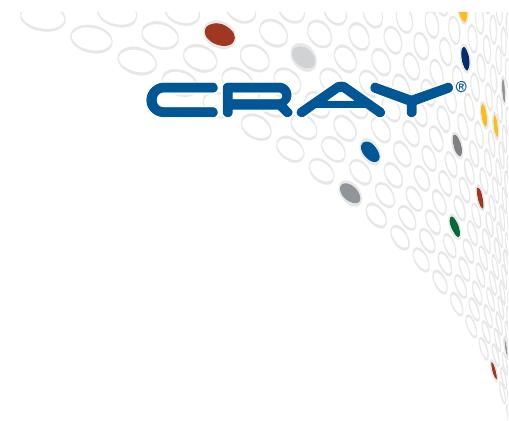
- **Two major releases per year** (April / October)
 - ~a month later: detailed [release notes](#)
 - latest release: Chapel 1.15, released April 6th 2017
- **CHIUW:** Chapel Implementers and Users Workshop (~June)
- **SC** (November)
 - tutorials, panels, BoFs, posters, educator sessions, exhibits, ...
 - annual **CHUG (Chapel Users Group)** happy hour
- **Talks, tutorials, research visits, blog posts, ...** (year-round)



A Year in the Life of Chapel

- Two major releases per year (April / October)
 - ~a month later: detailed release notes
 - latest release: Chapel 1.15, released April 6th 2017
- **CHIUW:** Chapel Implementers and Users Workshop (~June)
- **SC** (November)
 - tutorials, panels, BoFs, posters, educator sessions, exhibits, ...
 - annual **CHUG (Chapel Users Group)** happy hour
- Talks, tutorials, research visits, blog posts, ... (year-round)





Welcome to CHIUW 2017!

the 4th annual Chapel Implementers and Users Workshop



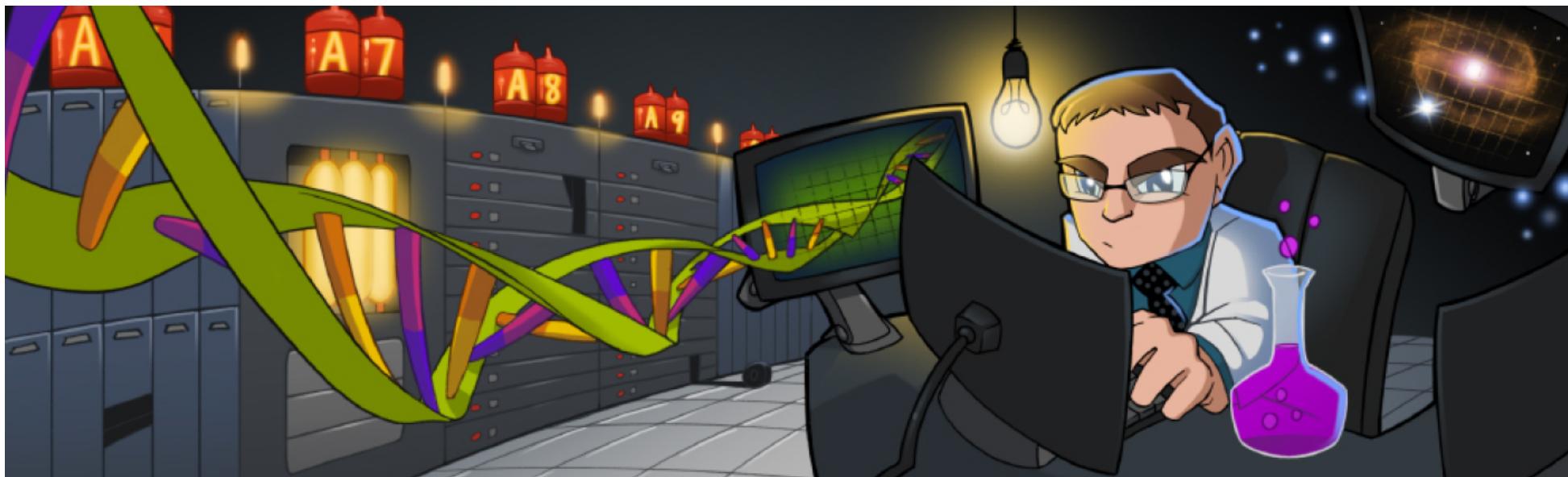
COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

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

Jonathan Dursi

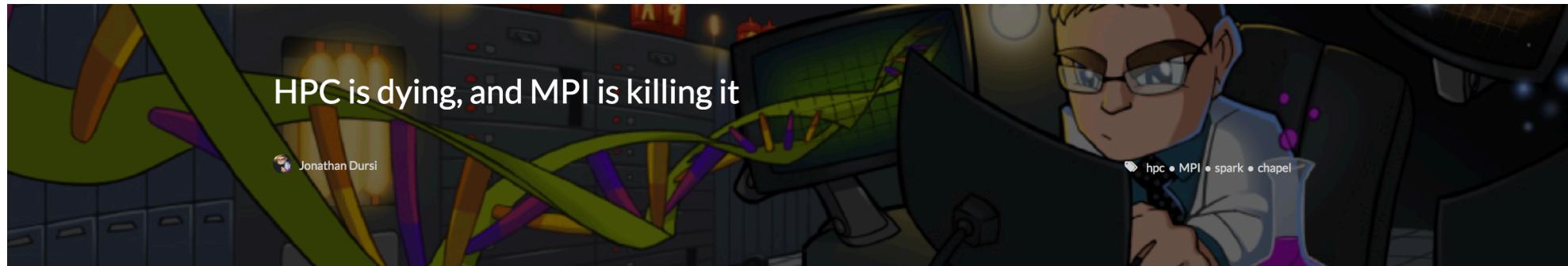
The Hospital for Sick Children, Toronto



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

CHIUW 2017: Keynote (“Why do I know that name...?”)



Jonathan Dursi

📍 Toronto



Pictured: The HPC community bravely holds off the incoming tide of new technologies and applications. Via [the BBC](#).

This should be a golden age for High Performance Computing.

For decades, the work of developing algorithms and implementations for tackling simulation and data analysis problems at the largest possible scales was obscure if important work. Then, suddenly, in the mid-2000s, two problems — analyzing internet-scale data, and interpreting an incoming flood of genomics data — arrived on the scene with data volumes and performance requirements which seemed quite familiar to HPCers, but with a size of audience unlike anything that had come before.

Suddenly discussions of topics of scalability, accuracy, large-scale data storage, and distributed matrix arithmetic all became mainstream and urgent. The number of projects and workshops addressing these topics exploded, and new energy went into implementing solutions problems faced in these domains.

In that environment, one might expect that programmers with HPC experience — who have dealt routinely with terabytes and now petabytes of data, and have years or decades of experience with designing and optimizing distributed memory algorithms — would be in high demand.



COMPUTE

STORE

ANALYZE

Copyright 2017 Cray Inc.

CHIUW 2017: Keynote (“Why do I know that name...?”)



Pictured: The HPC community bravely holds off the incoming tide of new technologies and applications. Via the BBC



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



CHIUW 2017: Research Papers

Identifying Use-After-Free Variables in Fire-and-Forget Tasks

Jyothi Krishna V S (*IIT Madras*) and Vassily Litvinov (*Cray Inc.*)

Towards a GraphBLAS Library in Chapel

Ariful Azad and Aydin Buluc (*LBNL*)

Comparative Performance and Optimization of Chapel in Modern Manycore Architectures

Engin Kayraklıoğlu, Wo Chang, and Tarek El-Ghazawi (*The George Washington University*)



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



CHIUW 2017: Technical Talks

Improving Chapel and Array Memory Management

Michael Ferguson, Vassily Litvinov, and Brad Chamberlain (*Cray Inc.*)

Try, Not Halt: An Error Handling Strategy for Chapel

Preston Sahabu, Michael Ferguson, Greg Titus, and Kyle Brady (*Cray Inc.*)

GPGPU support in Chapel with the Radeon Open Compute Platform

Michael Chu, Ashwin Aji, Daniel Lowell, and Khaled Hamidouche (*AMD*)

An OFI libfabric Communication Layer for the Chapel Runtime

Sung-Eun Choi (*Cray Inc.*)

Sketching Streams with Chapel

Christopher Taylor (*DoD*)

Entering the Fray: Chapel's Computer Language Benchmarks Game Entry

Brad Chamberlain, Ben Albrecht, Lydia Duncan, Ben Harshbarger, Elliot Ronaghan, Preston Sahabu, Michael Noakes (*Cray Inc.*), and Laura Delaney (*Whitworth University*)





CHIUW 2017: Planning Committee

General Chairs:

- Tom MacDonald, *Cray Inc.*
- Michael Ferguson, *Cray Inc.*

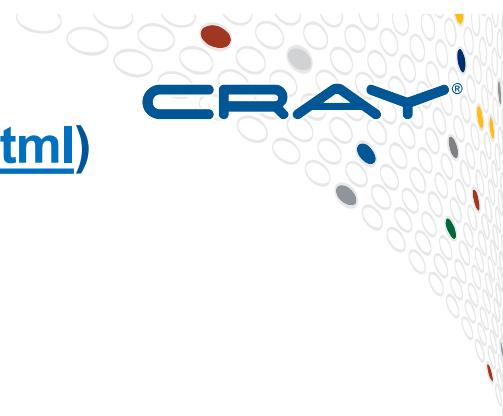
Program Committee:

- Brad Chamberlain (chair), *Cray Inc.*
- Nikhil Padmanabhan (co-chair), *Yale University*
- Richard Barrett, *Sandia National Laboratories*
- Mike Chu, *AMD*
- Mary Hall, *University of Utah*
- Jeff Hammond, *Intel*
- Jeff Hollingsworth, *University of Maryland*
- Cosmin Oancea, *University of Copenhagen*
- David Richards, *Lawrence Livermore National Laboratory*
- Michelle Strout, *University of Arizona*
- Kenjiro Taura, *University of Tokyo*



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



CHIUIW 2017: Agenda (chapel.cray.com/CHIUW2017.html)

- 8:30: Chapel Boot Camp (optional)
- 9:00: **Welcome, State of the Project**
- 9:30: **Break**
- 10:00: **Talks: Chapel Design and Implementation**
- 11:10: **Quick Break**
- 11:20: **Talks: Targeting New Architectures**
- 12:00: **Lunch**
- 1:30: **Keynote Talk: Jonathan Dursi**
- 2:30: **Talks: Uses of Chapel**
- 3:20: **Break**
- 3:50: **Talks: Benchmarking and Performance**
- 4:40: **Lightning Talks and Flash Discussions**
- 5:30: **Wrap-up / Head to Dinner**



COMPUTE

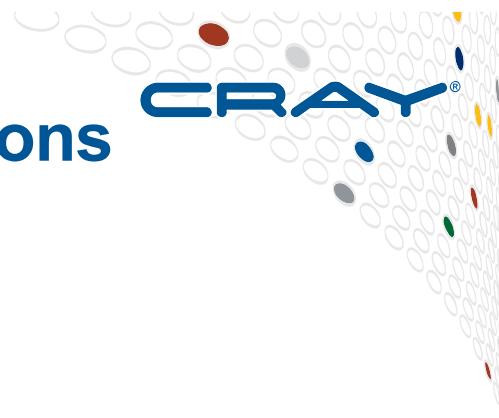
|

STORE

|

ANALYZE

CHIUW 2017: Lightning Talks & Flash Discussions



- New this year!
- Last session of the day!
- Goal: high-energy hot topics for low attention spans!
- Format: Short talks, Q&A, war stories, ...whatever!
- Sign up for a slot!



COMPUTE

|

STORE

|

ANALYZE

Copyright 2017 Cray Inc.

CHIUW 2017: Lightning Talks & Flash Discussions



- New this year!
- Last session of the day!
- Goal: high-energy hot topics for low attention spans!
- Format: Short talks, Q&A, war stories, ...whatever!
- Sign up for a slot!



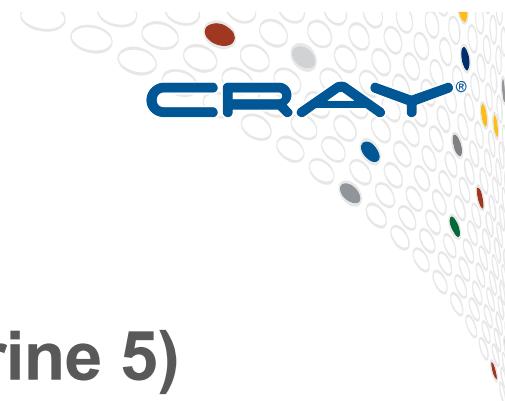
COMPUTE

|

STORE

|

ANALYZE



CHIUIW 2017: Code Camp (Half) Day

- Tomorrow morning: 8:30-noon (room: Tangerine 5)
- Proposed topics (so far):
 - write a user-defined domain map
 - Chapel on AWS
 - work on GPU support
 - introduction to Chapel code generation & optimizations
 - network atomics in Chapel's runtime libraries
 - re-architect launcher scripts
 - unified communication diagnostics hooks
 - storage technologies
 - beef up Linear Algebra module
 - port gravitational n-body code to Chapel
 - pyChapel improvements
 - ...
 - **your idea here?**



COMPUTE

|

STORE

|

ANALYZE



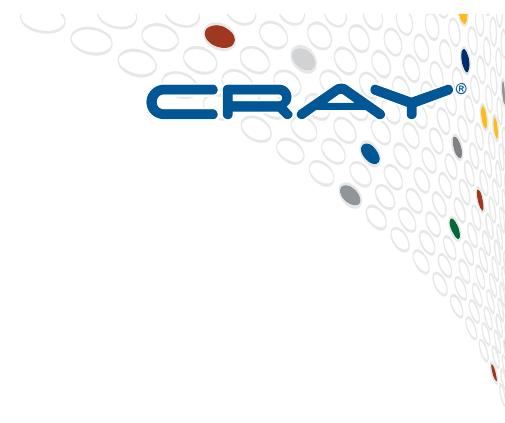
SWAG and Surveys

- **We have a few giveaways today:**
 - Chapel stickers
 - Chapel microfiber wipes for screens / glasses
- **We also have a CHIUW survey**
 - available in paper or online form—please fill one out



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



State of the Chapel Project 2017



COMPUTE

|

STORE

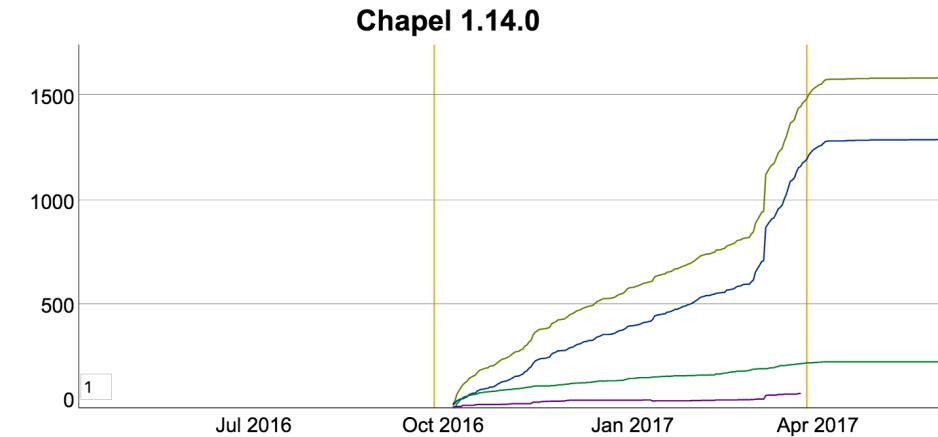
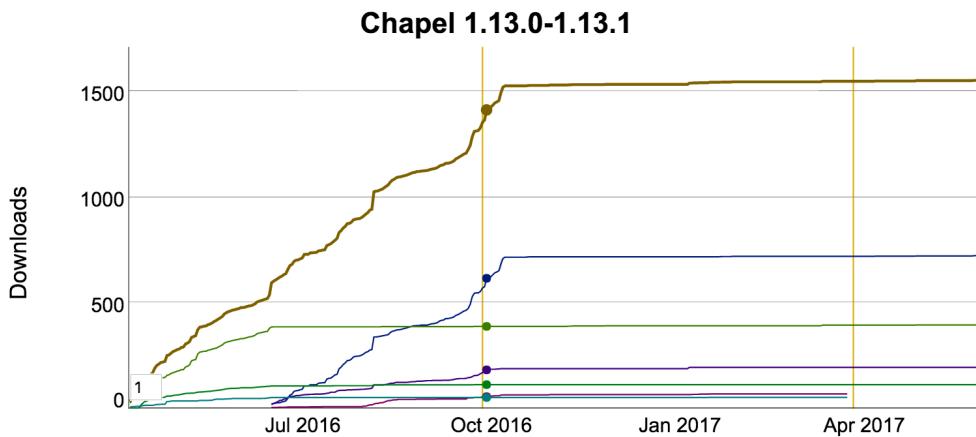
|

ANALYZE

Copyright 2017 Cray Inc.

Releases since CHIUW 2016

- Since last year, two new major versions of Chapel:
 - 1.14: October 6, 2016
 - 1.15: April 6, 2017 ← our most significant release ever!
- Significant progress in all areas of the release
 - performance, memory leaks, libraries, documentation, portability, ...
- Achieving 1500+ downloads per release





Contributors to the Past Year's Releases

Contributors to 1.14 / 1.15:

- Ben Albrecht, Cray Inc.
- Matthew Baker, ORNL
- Paul Cassella, Cray Inc.
- Brad Chamberlain, Cray Inc.
- Sung-Eun Choi, Cray Inc.
- Marcos Cleison Silva Santana, individual contributor
- Laura Delaney, Whitworth University / Cray
- Lydia Duncan, Cray Inc.
- Michael Ferguson, Cray Inc.
- Ben Harshbarger, Cray Inc.
- Andrea Francesco Iuorio, Università degli Studi di Milano / GSoC
- David Iten, Cray Inc.
- David Keaton, Cray Inc.
- Engin Kayraklioglu, George Washington University / Cray Inc.
- Sagar Khatri, individual contributor
- Przemysław Leśniak, individual contributor

- Vassily Litvinov, Cray Inc.
- Tom MacDonald, Cray Inc.
- Deepak Majeti, individual contributor
- Phil Nelson, Western Washington University / Cray
- Michael Noakes, Cray Inc.
- Nikhil Padmanabhan, Yale University
- Nicholas Park, DOD
- Sriraj Paul, Rice University
- Kumar Prasun, individual contributor
- Elliot Ronaghan, Cray Inc.
- Preston Sahabu, Cray Inc.
- Kushal Singh, Int'l Institute of Information Technology, Hyderabad / GSoC
- Kenjiro Taura, University of Tokyo
- Chris Taylor, DOD
- Greg Titus, Cray Inc.
- Rob Upcraft, individual contributor
- Tony Wallace, Cray Inc.
- Hui Zhang, [University of Maryland]

This year saw a record number of contributors to the releases





Contributors to the Past Year's Releases

Contributors to 1.14 / 1.15:

- Ben Albrecht, Cray Inc.
- Matthew Baker, ORNL
- Paul Cassella, Cray Inc.
- Brad Chamberlain, Cray Inc.
- Sung-Eun Choi, Cray Inc.
- Marcos Cleison Silva Santana, individual contributor
- Laura Delaney, Whitworth University / Cray
- Lydia Duncan, Cray Inc.
- Michael Ferguson, Cray Inc.
- Ben Harshbarger, Cray Inc.
- Andrea Francesco Iuorio, Università degli Studi di Milano / GSoC
- David Iten, Cray Inc.
- David Keaton, Cray Inc.
- Engin Kayraklioglu, George Washington University / Cray Inc.
- Sagar Khatri, individual contributor
- Przemysław Leśniak, individual contributor

- Vassily Litvinov, Cray Inc.
- Tom MacDonald, Cray Inc.
- Deepak Majeti, individual contributor
- Phil Nelson, Western Washington University / Cray
- Michael Noakes, Cray Inc.
- Nikhil Padmanabhan, Yale University
- Nicholas Park, DOD
- Sriraj Paul, Rice University
- Kumar Prasun, individual contributor
- Elliot Ronaghan, Cray Inc.
- Preston Sahabu, Cray Inc.
- Kushal Singh, Int'l Institute of Information Technology, Hyderabad / GSoC
- Kenjiro Taura, University of Tokyo
- Chris Taylor, DOD
- Greg Titus, Cray Inc.
- Rob Upcraft, individual contributor
- Tony Wallace, Cray Inc.
- Hui Zhang, [University of Maryland]

17 Cray employees, 3 Cray summer interns/contractors, 15 external contributors



The Chapel Team at Cray (May 2017)



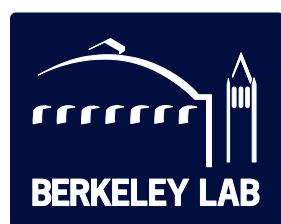
COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

Chapel R&D Organizations



THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC



Lawrence Berkeley
National Laboratory



Yale

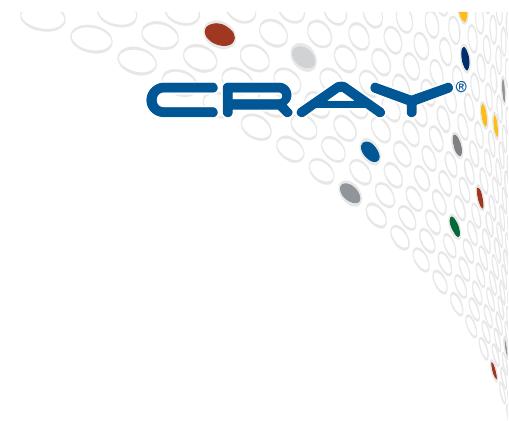
(and several others...)

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



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



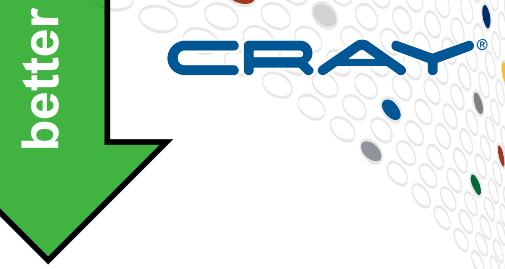
Single-Locale Performance



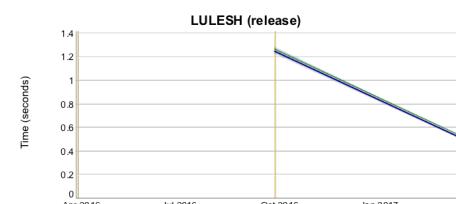
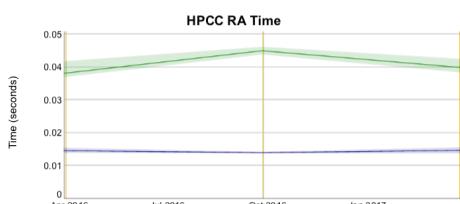
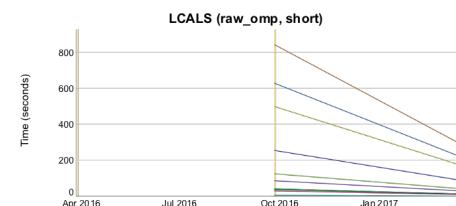
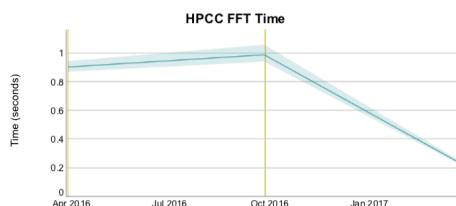
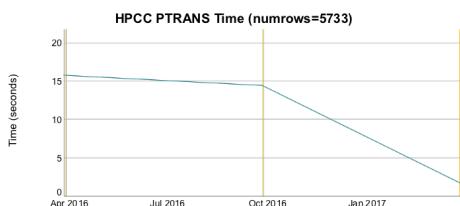
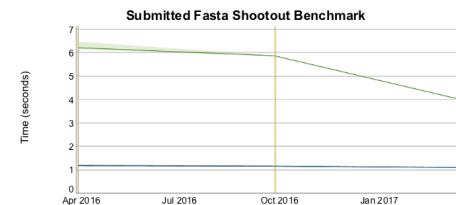
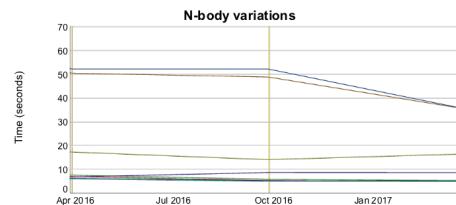
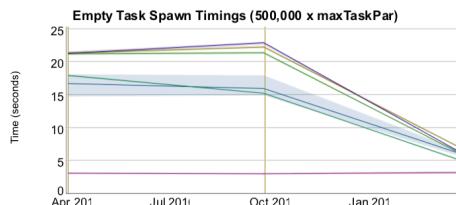
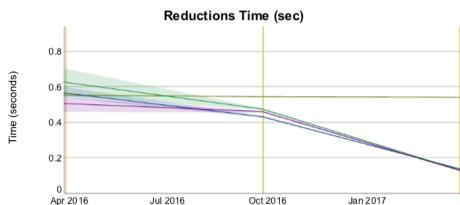
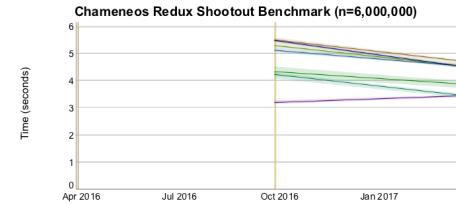
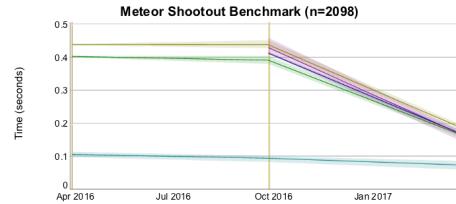
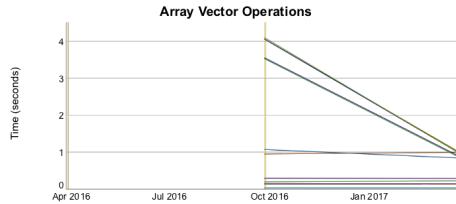
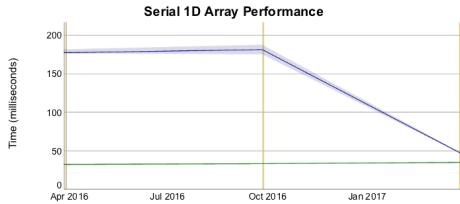
COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

Single-Locale Performance: the past year



- Overall, single-locale performance improved dramatically



COMPUTE

STORE

ANALYZE

Copyright 2017 Cray Inc.

LCALS: Serial Timings, Chapel 1.13.0

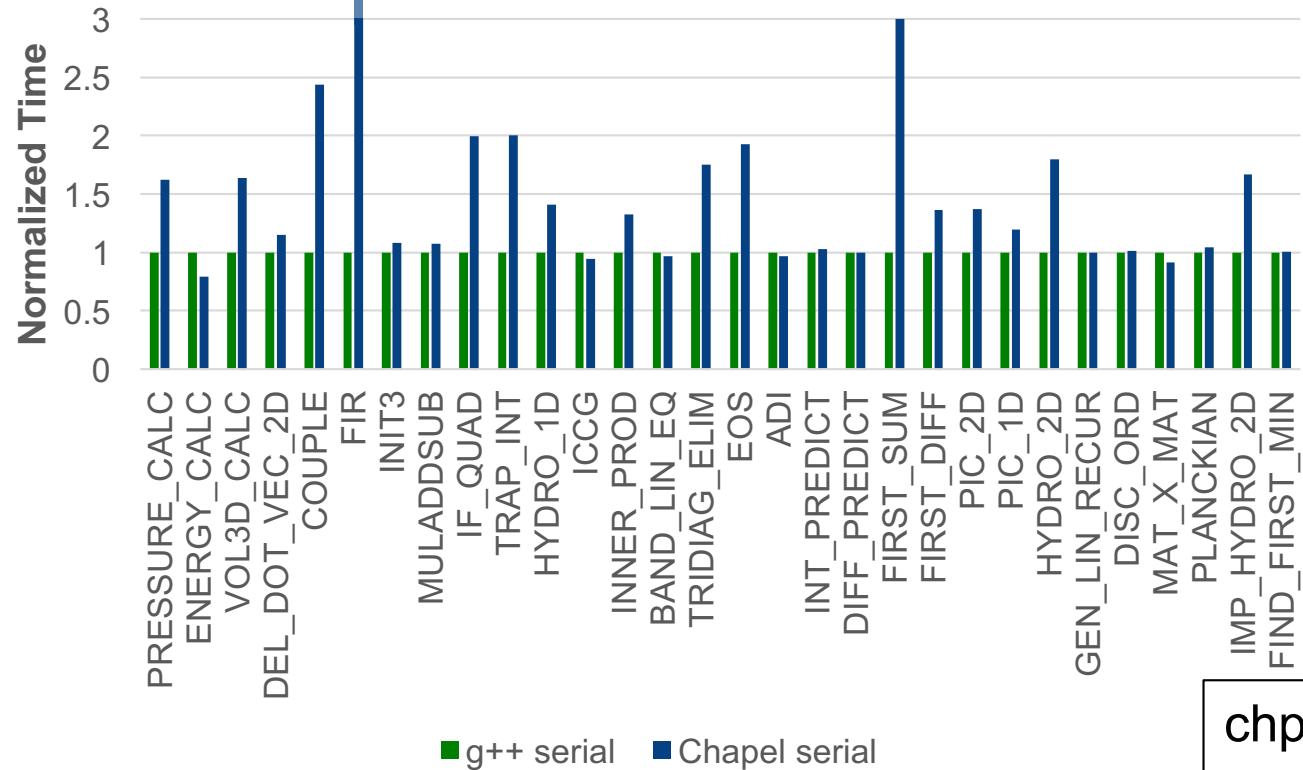


Long problem size

(Similar results for medium
and short problem sizes)

Serial Chapel vs g++

Normalized time –
serial reference is 1.0



better

chpl --fast
--no-ieee-float

g++ -Ofast -fopenmp



COMPUTE

STORE

ANALYZE

Copyright 2017 Cray Inc.

LCALS: Serial Timings, Chapel 1.14.0

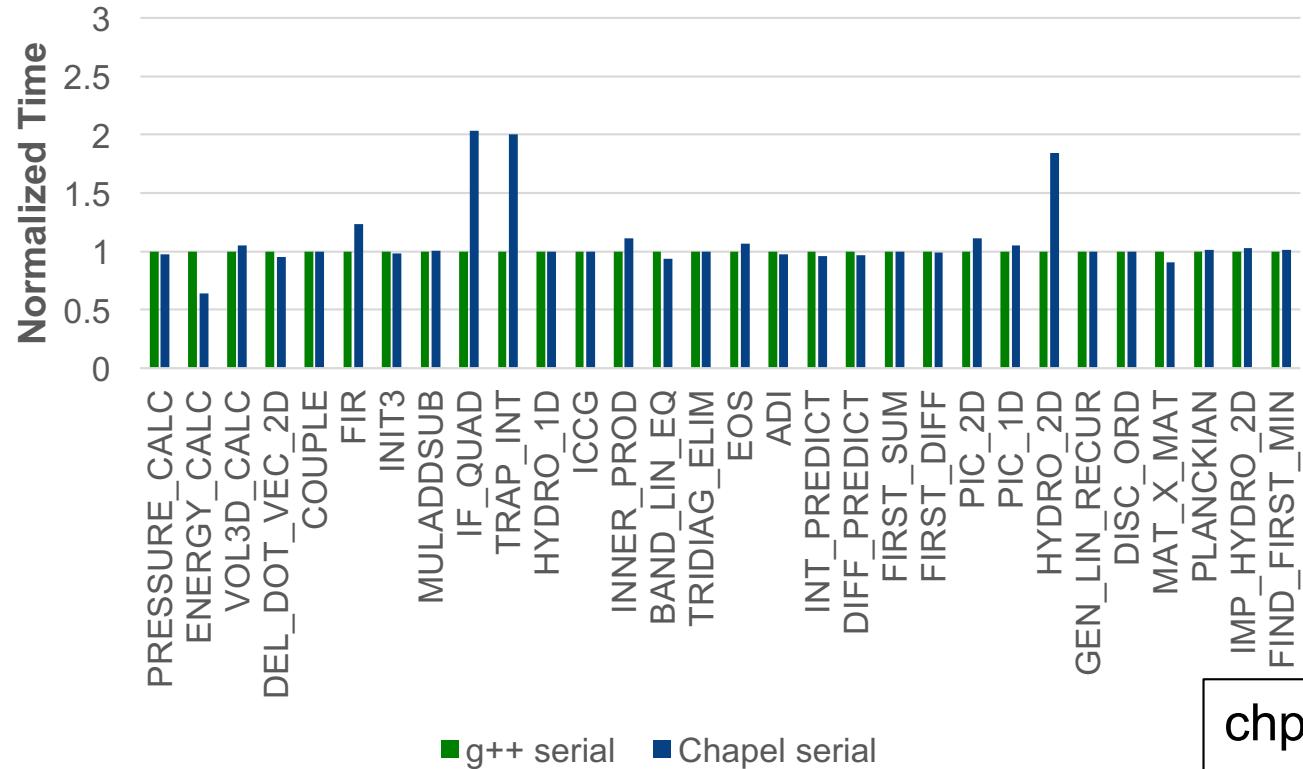


Long problem size

(Similar results for medium
and short problem sizes)

Serial Chapel vs g++

Normalized time –
serial reference is 1.0



chpl --fast
--no-ieee-float

g++ -Ofast -fopenmp



COMPUTE

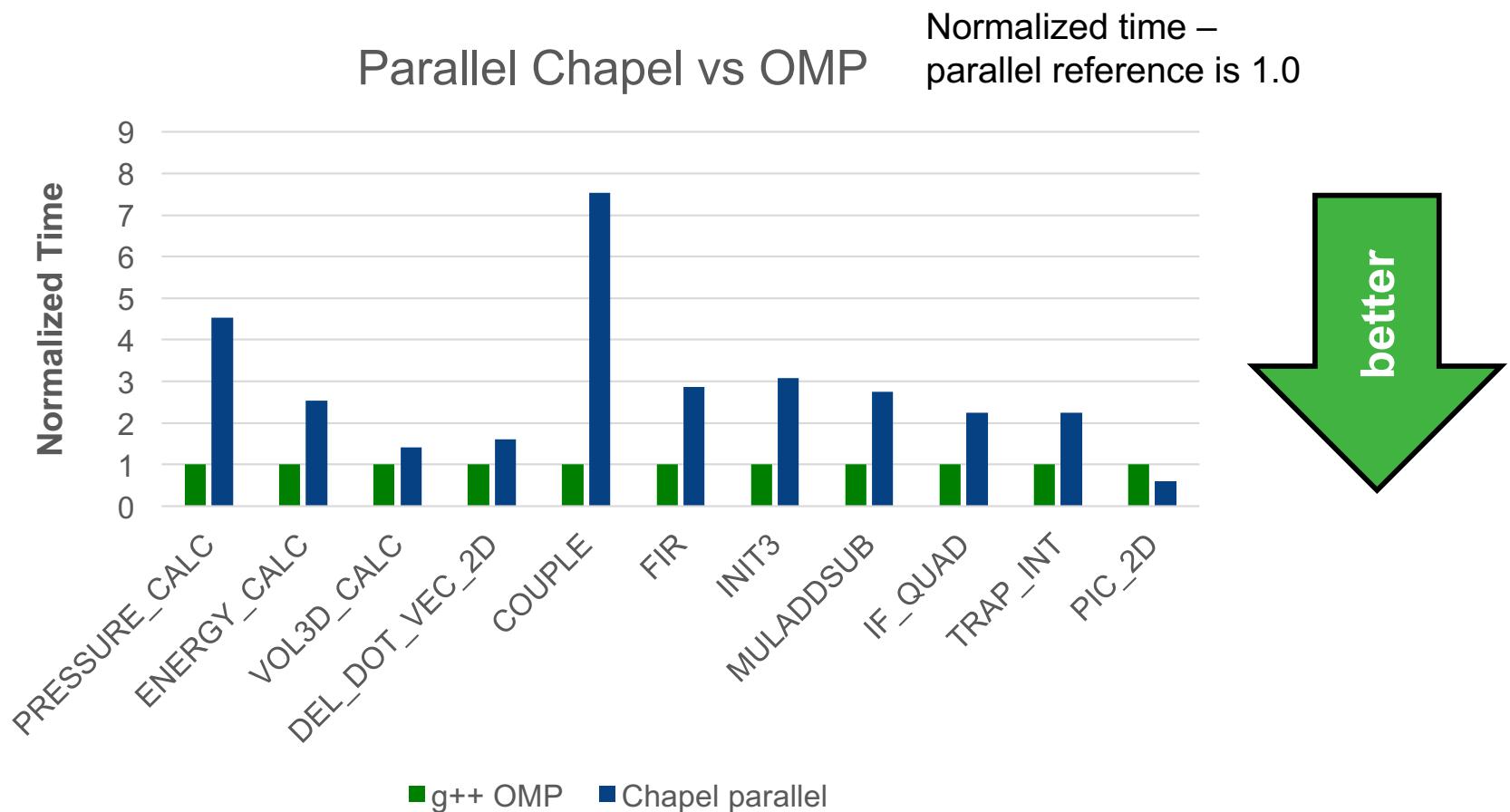
STORE

ANALYZE

LCALS: Parallel Timings, Chapel 1.14.0



- Parallel variants still lagged behind the reference in 1.14
 - between 1.5X and 8X slower for long problem size



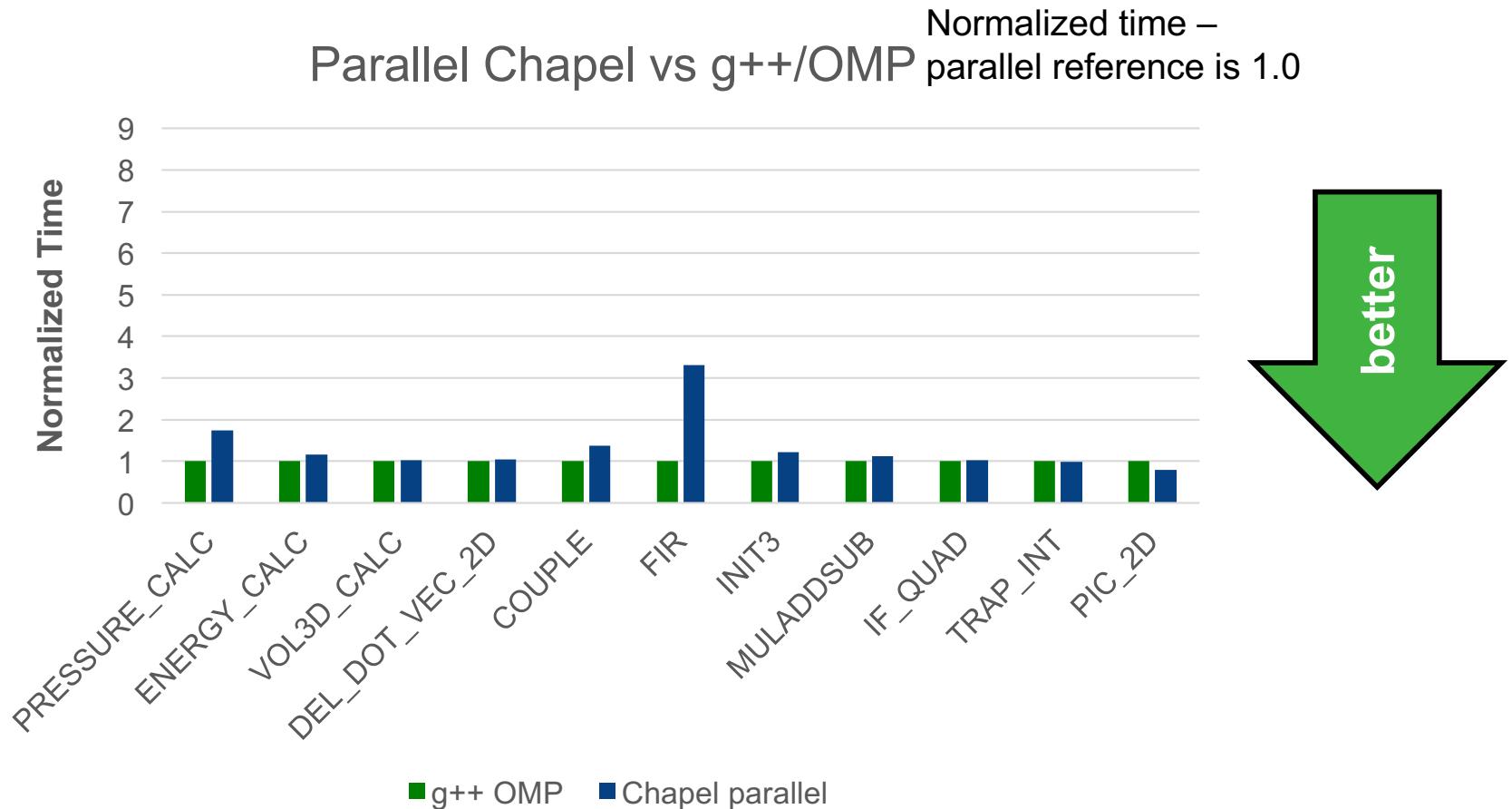
COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

LCALS: Parallel Timings, Chapel 1.15.0



- Chapel 1.15 closed the gap significantly
 - ~3-4x speedup: on par or very close to reference for most kernels



COMPUTE

STORE

ANALYZE



The Computer Language Benchmarks Game

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 fast?

Which are succinct? Which are efficient?

Ada C Chapel C# C++ Dart

Erlang F# Fortran Go Hack

Haskell Java JavaScript Lisp Lua

OCaml Pascal Perl PHP Python

Racket Ruby JRuby Rust Smalltalk

Swift TypeScript

{ for researchers } fast-faster-fastest

stories

Since CHI UW 2016,
Chapel entry completed
and listed on site

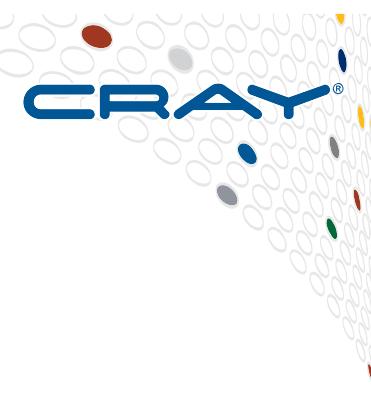


COMPUTE

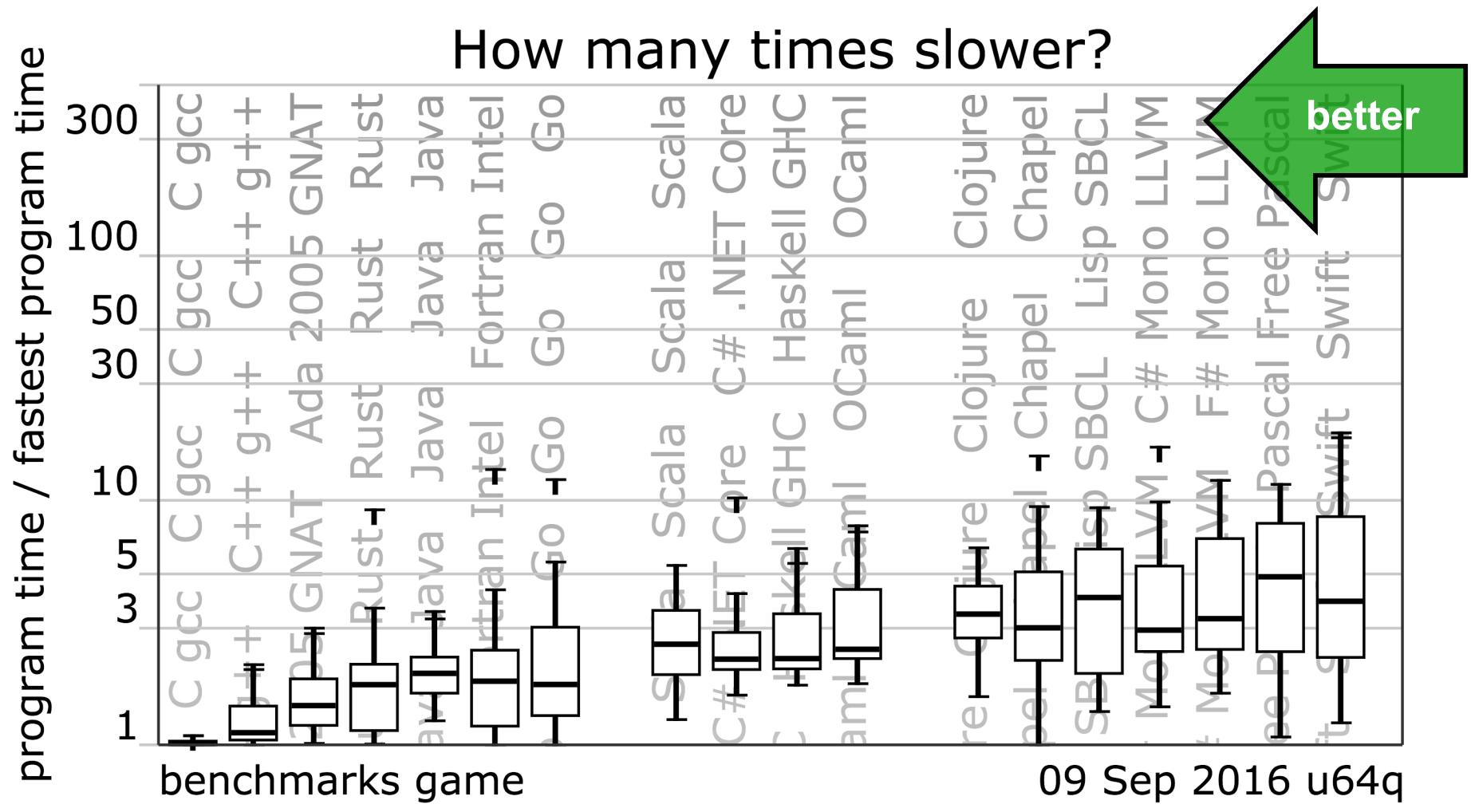
STORE

ANALYZE

CLBG: Fast-faster-fastest graph (Sep 2016)



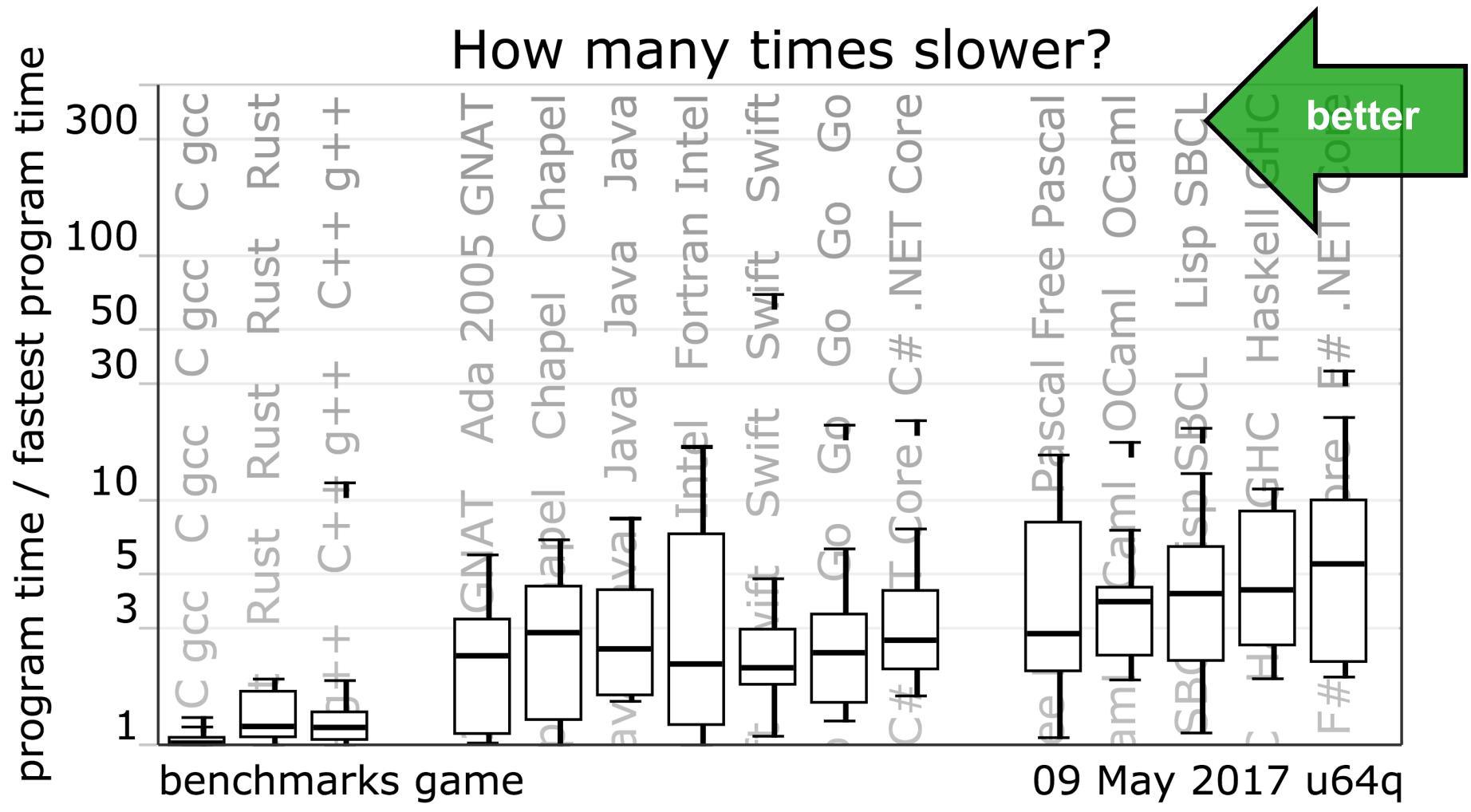
Relative performance, sorted by geometric mean



CLBG: Fast-faster-fastest graph (May 2017)



Relative performance, sorted by geometric mean



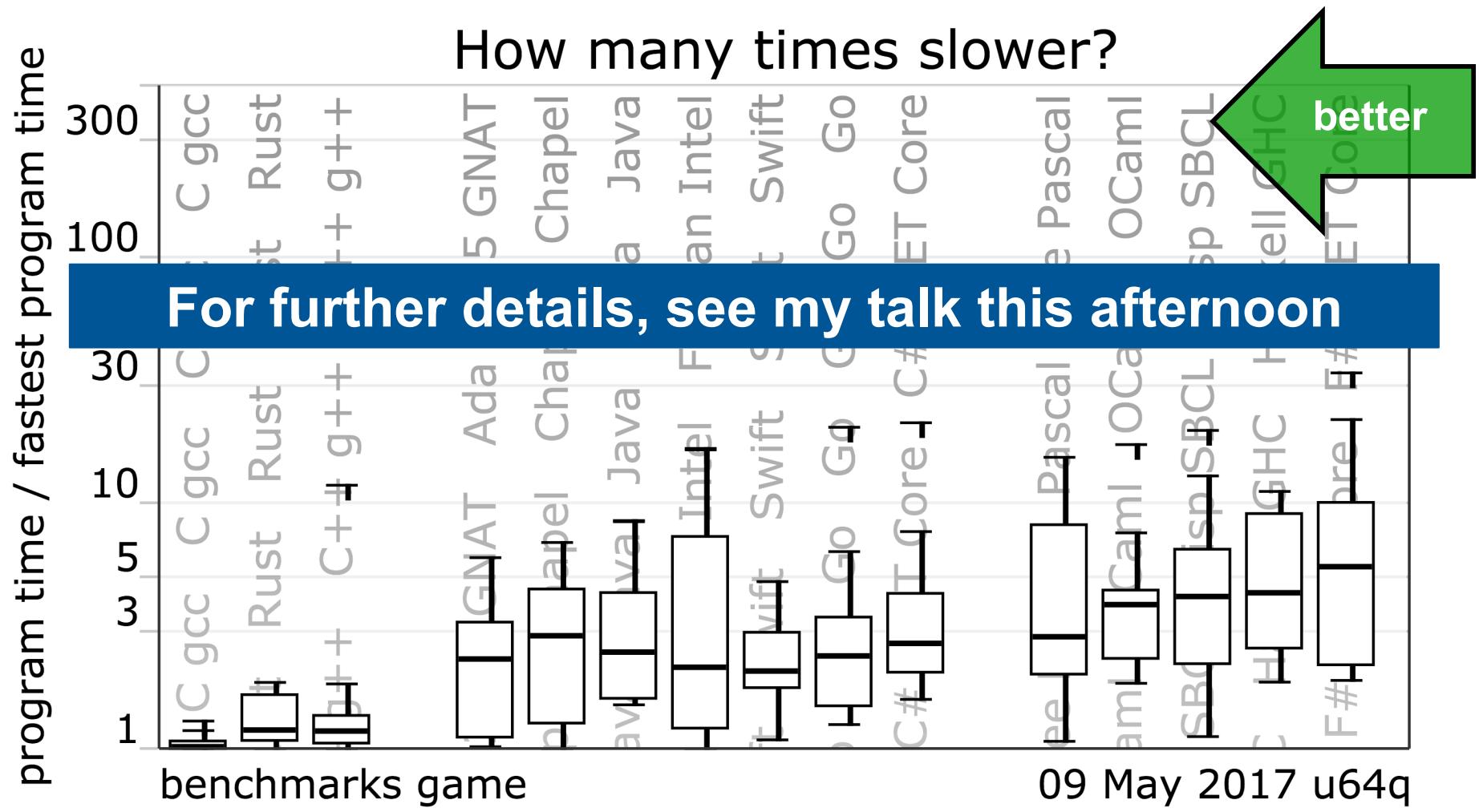
COMPUTE | STORE | ANALYZE

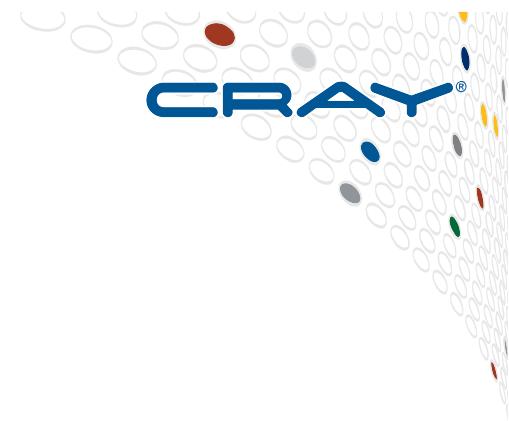
Copyright 2017 Cray Inc.

CLBG: Fast-faster-fastest graph (May 2017)



Relative performance, sorted by geometric mean





Multi-Locale Performance



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



PGAS Applications Workshop

Monday, November 14th, 2016

Held in conjunction with SC16



In cooperation with:



Optimizing PGAS overhead in a multi-locale Chapel implementation of CoMD

Riyaz Haque and David F. Richards



LLNL-PRES-708978

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

 Lawrence Livermore
National Laboratory



Copyright 2017 Cray Inc.

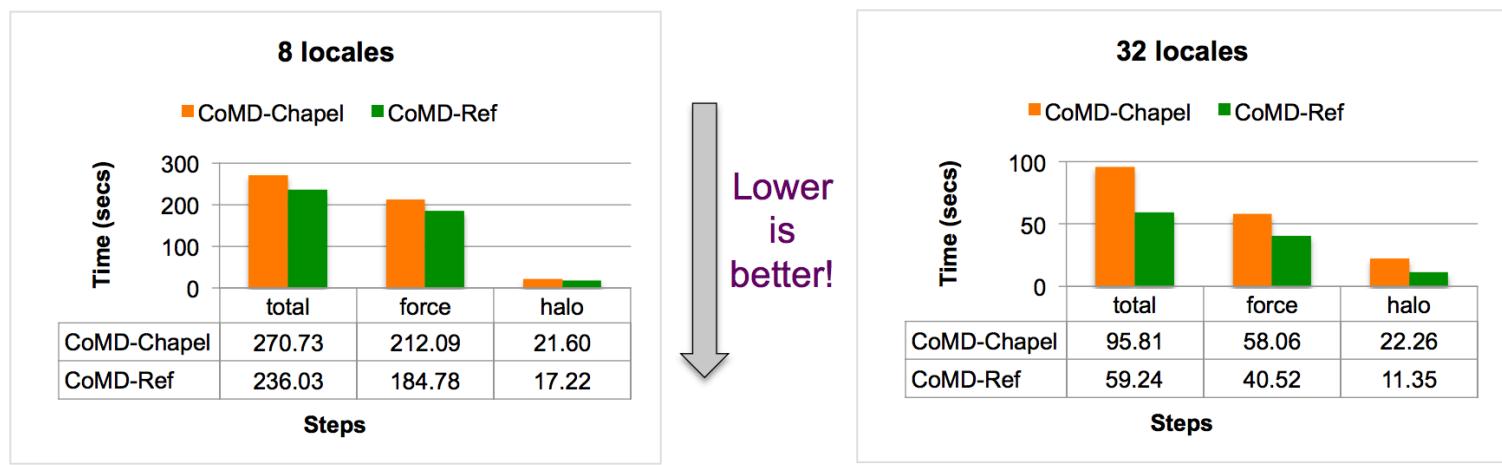
Opt Chap



LLNL-PRES-708978
This work was performed under contract DE-AC52-07NA2-

Performance is comparable to the reference implementation

- Code compiled using Chapel v1.13
 - The compiler itself was compiled with gcc-4.9.2, ibv on gasnet and qthreads as the threading framework
- Executed on 1-32 nodes of 64-bit Intel Xeon processors
 - 12 cores, 24 GB RAM, Infiniband high-speed interconnect

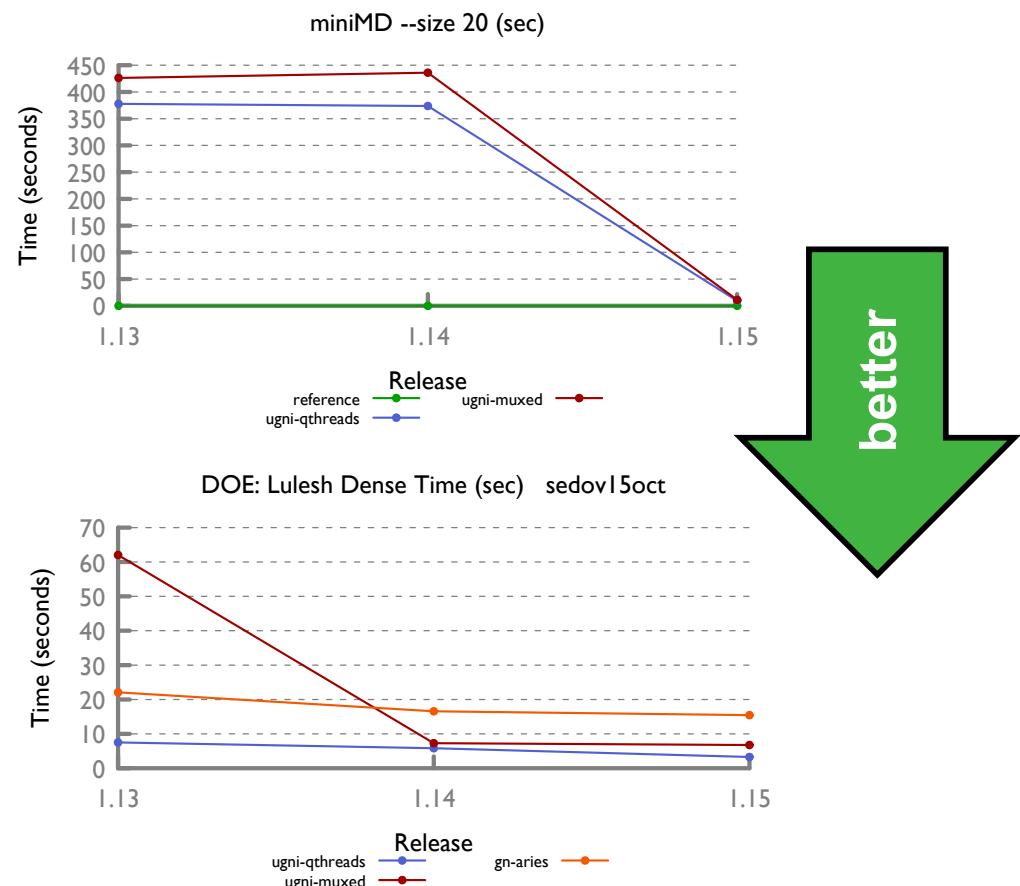
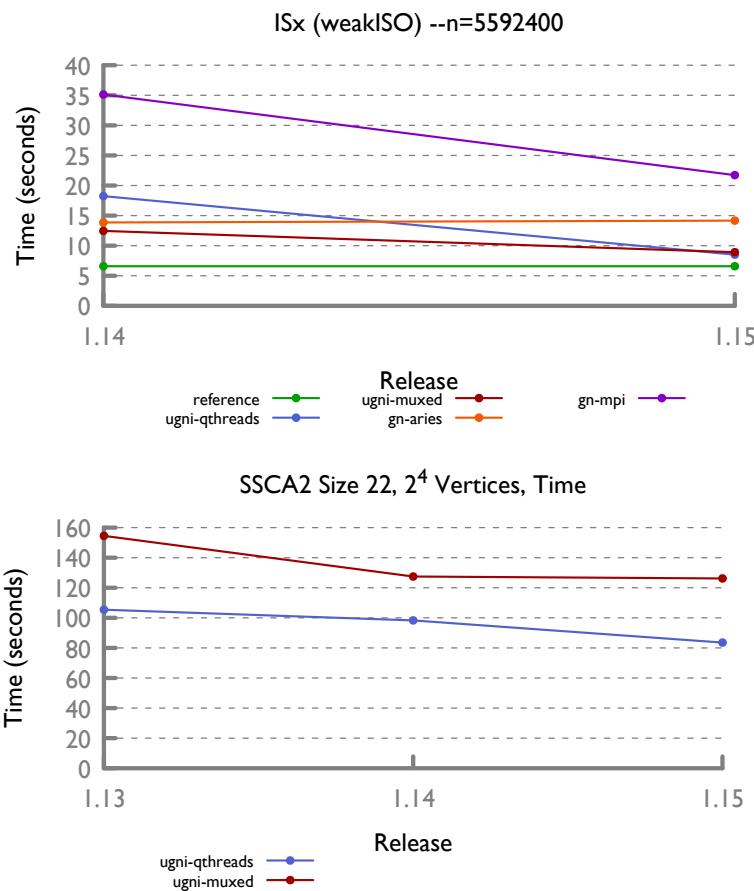


CoMD-Chapel performs to within 87% (8 locales) to 67% (32 locales) of the reference



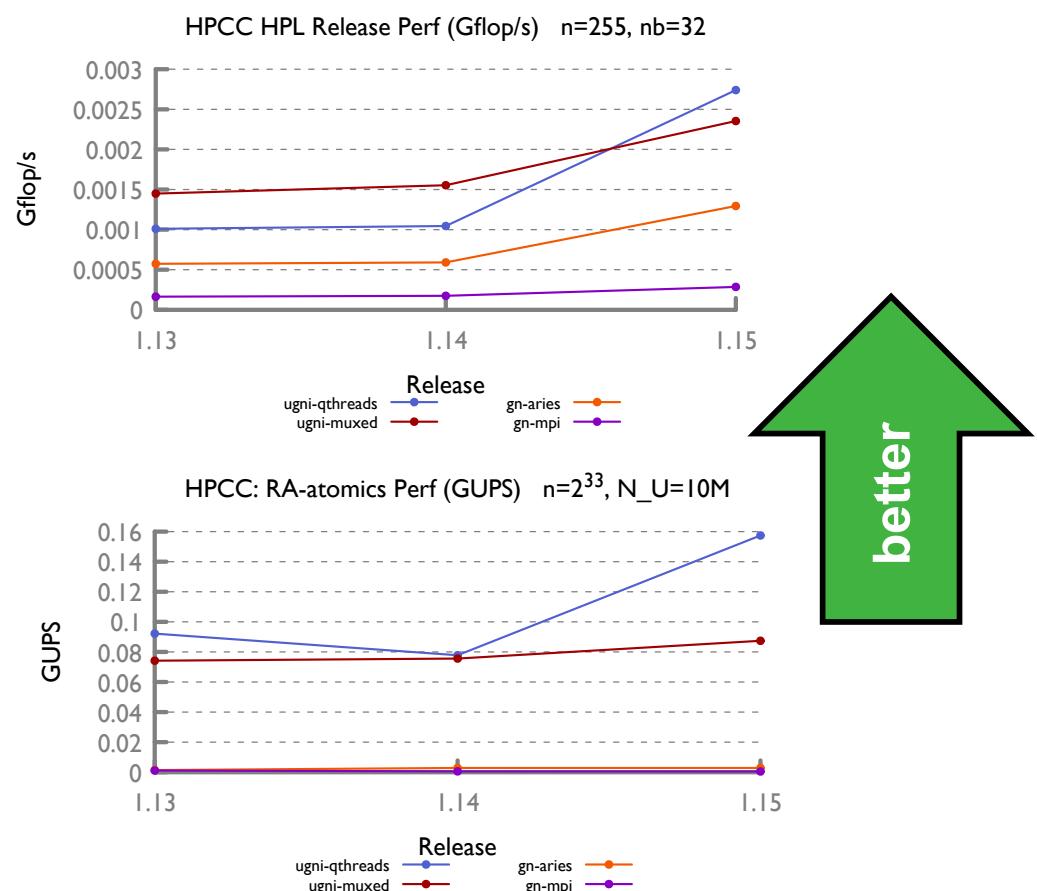
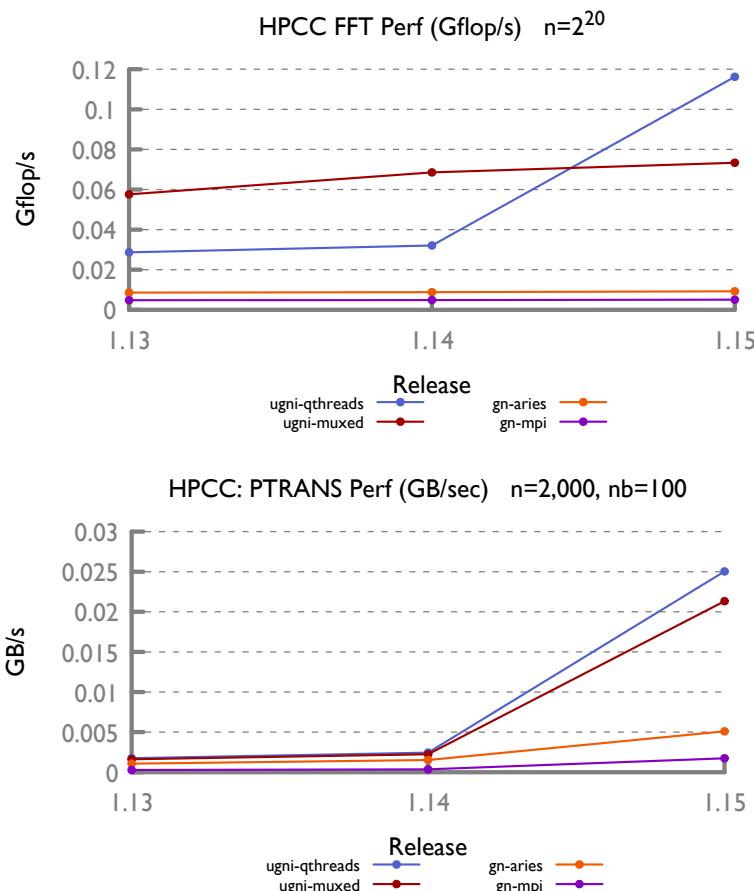
Multi-locale Performance

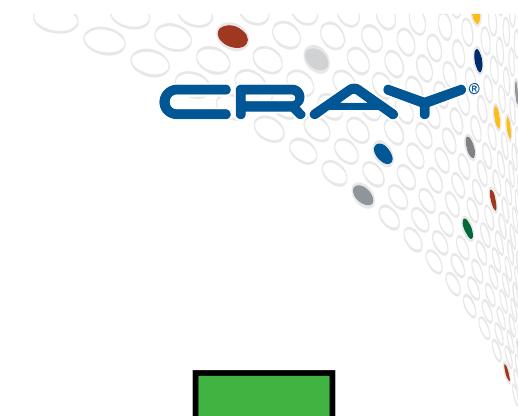
- Significant multi-locale performance improvements
 - no known regressions



Multi-locale Performance

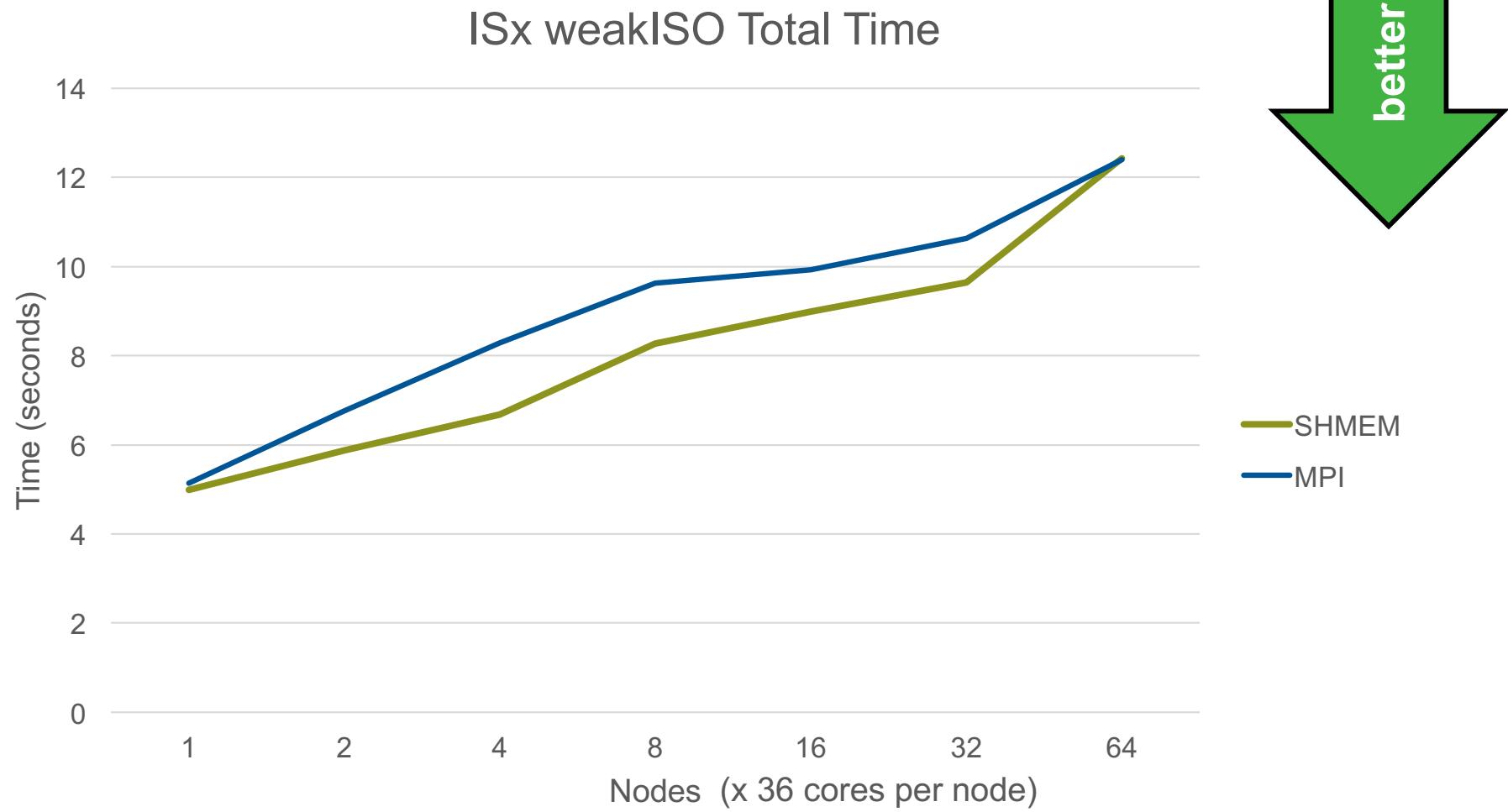
- Significant multi-locale performance improvements
 - no known regressions (qthreads now outperforms muxed even more)





ISx Execution Time: MPI, SHMEM

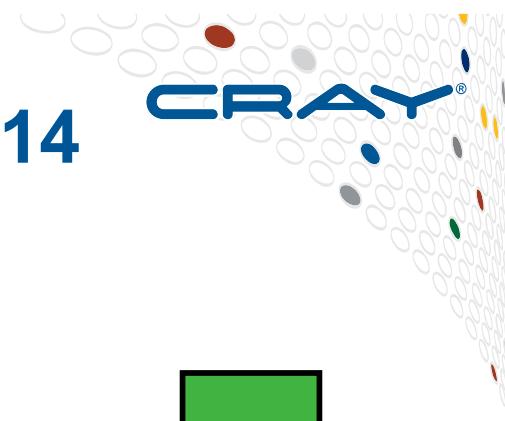
- 64 nodes on Cray XC



COMPUTE

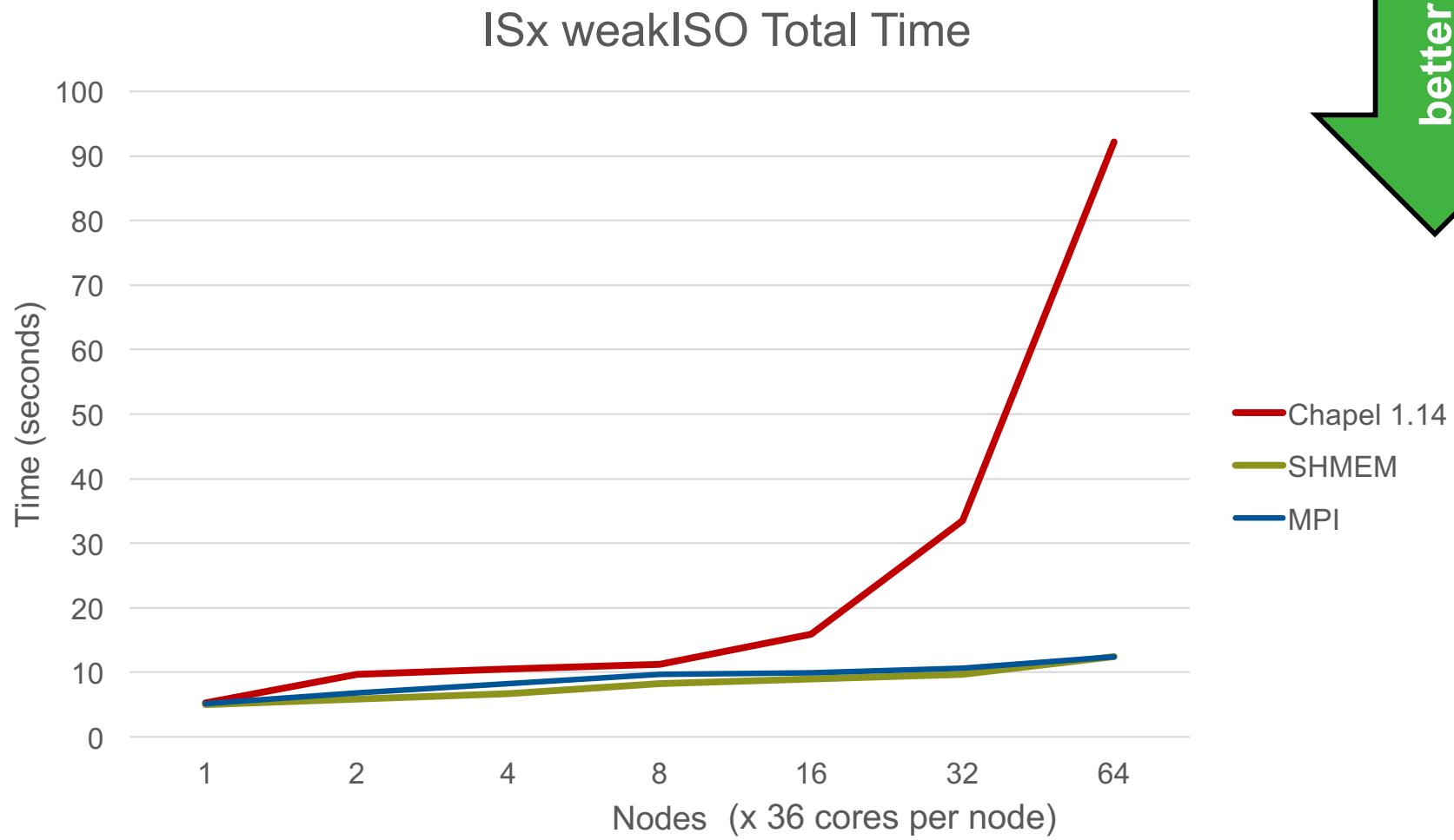
STORE

ANALYZE



ISx Execution Time: MPI, SHMEM, Chapel 1.14

- 64 nodes on Cray XC



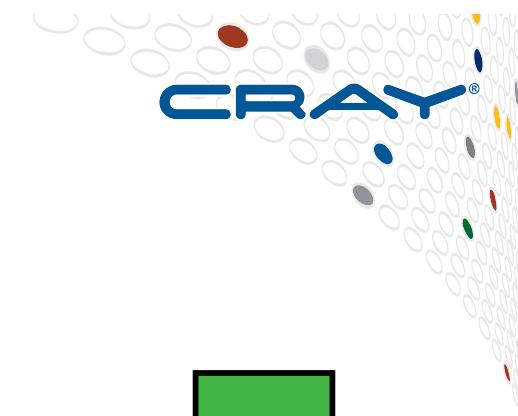
better



COMPUTE

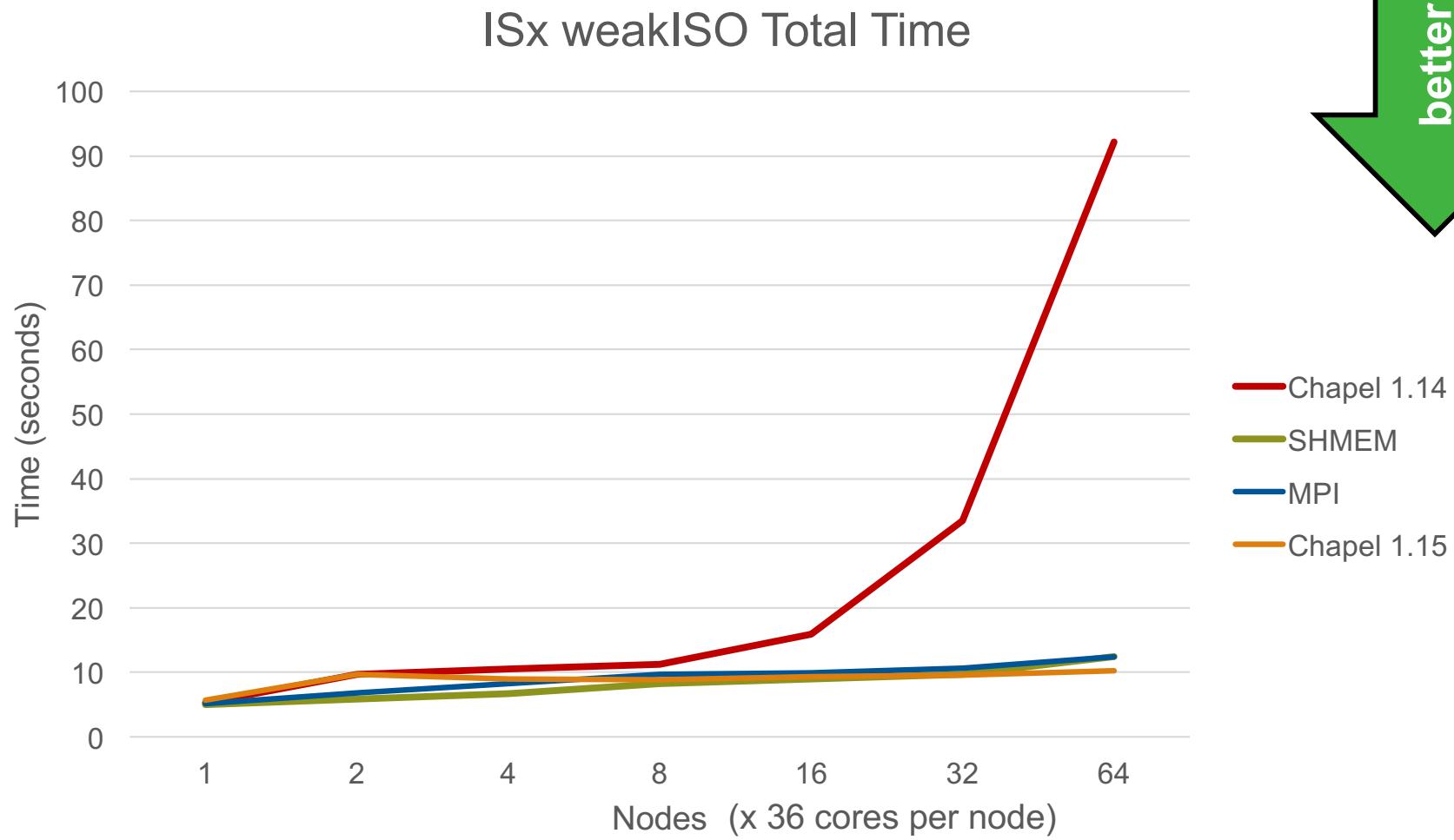
STORE

ANALYZE



ISx Execution Time: MPI, SHMEM, Chapel

- 64 nodes on Cray XC



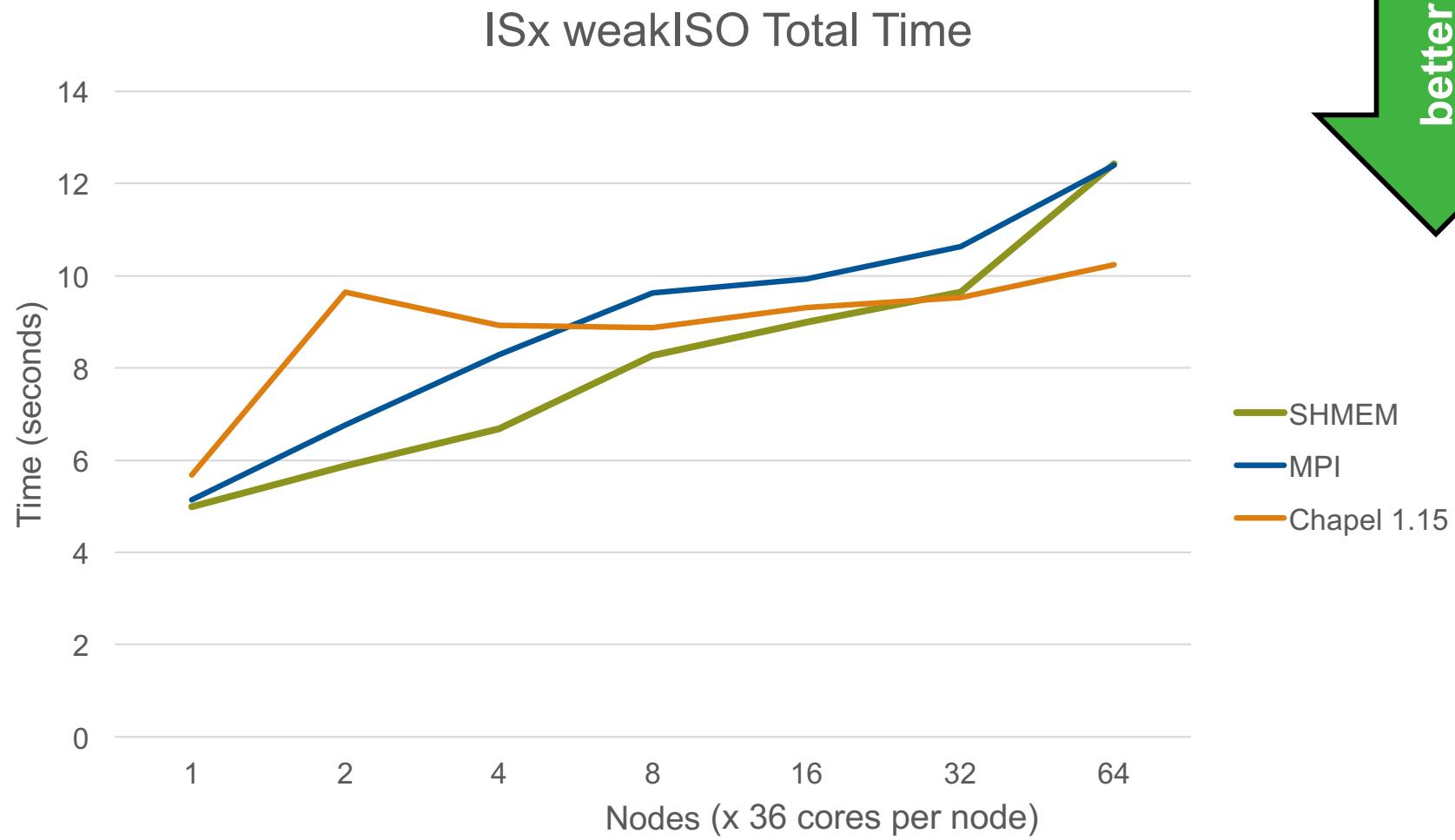
COMPUTE

STORE

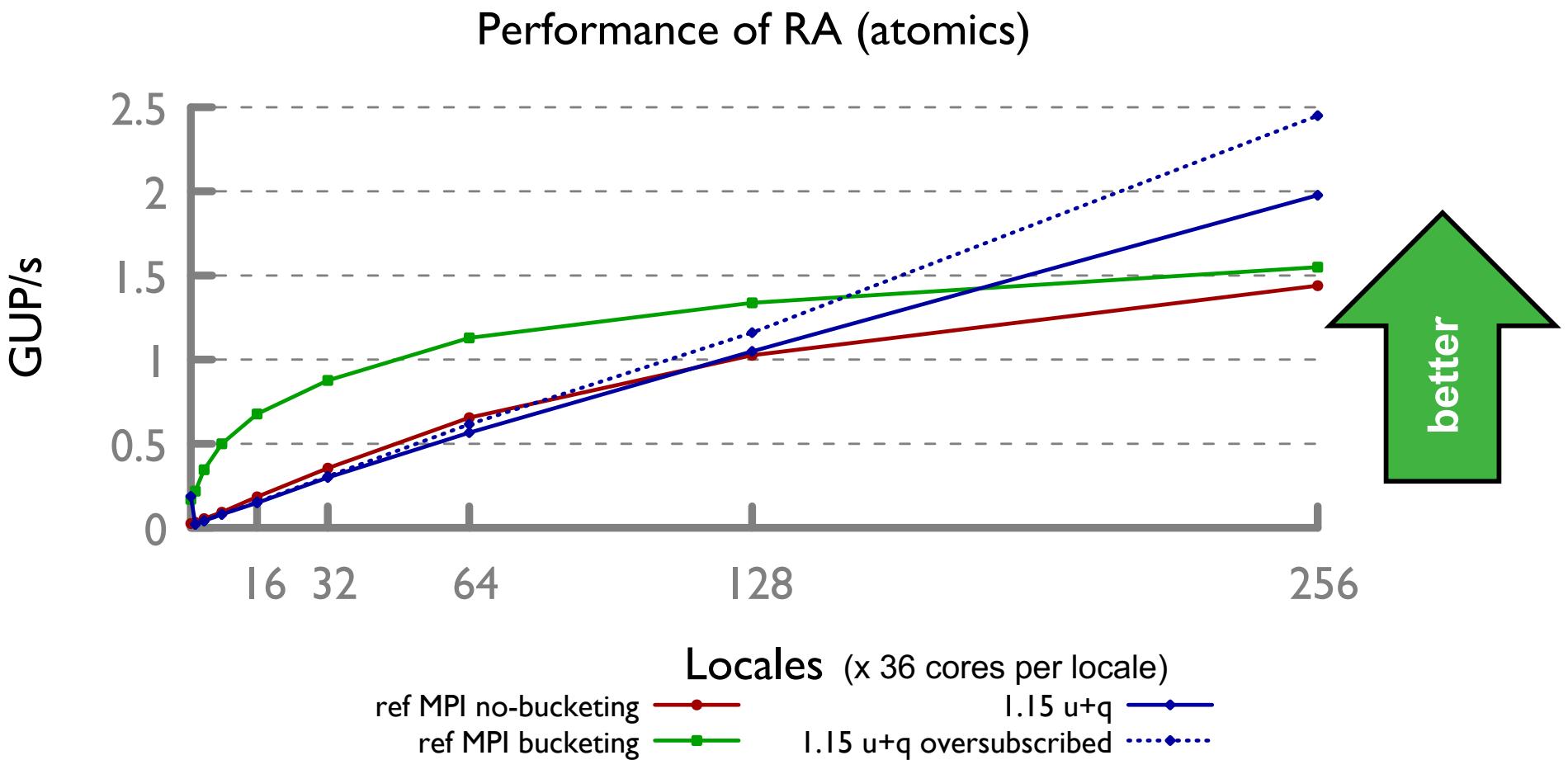
ANALYZE

ISx Execution Time: MPI, SHMEM, Chapel 1.15

- 64 nodes on Cray XC



RA Performance: Chapel vs. MPI



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Performance: Summary

Summary:

- Chapel has achieved dramatic performance improvements this year

Next steps:

- **Multi-locale:**
 - benchmark-driven performance and scalability improvements
 - particularly for stencils, PRKs, motivating applications
- **Single-locale:**
 - vectorization



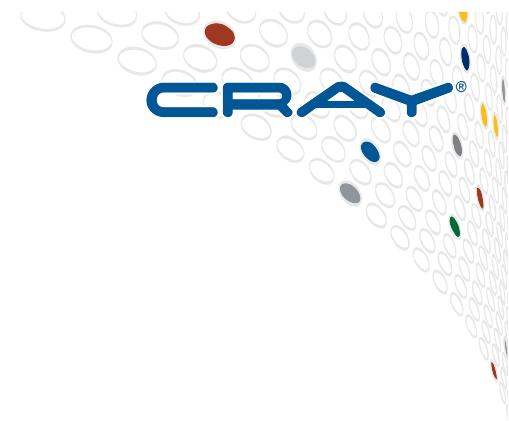
COMPUTE

|

STORE

|

ANALYZE



Memory Improvements



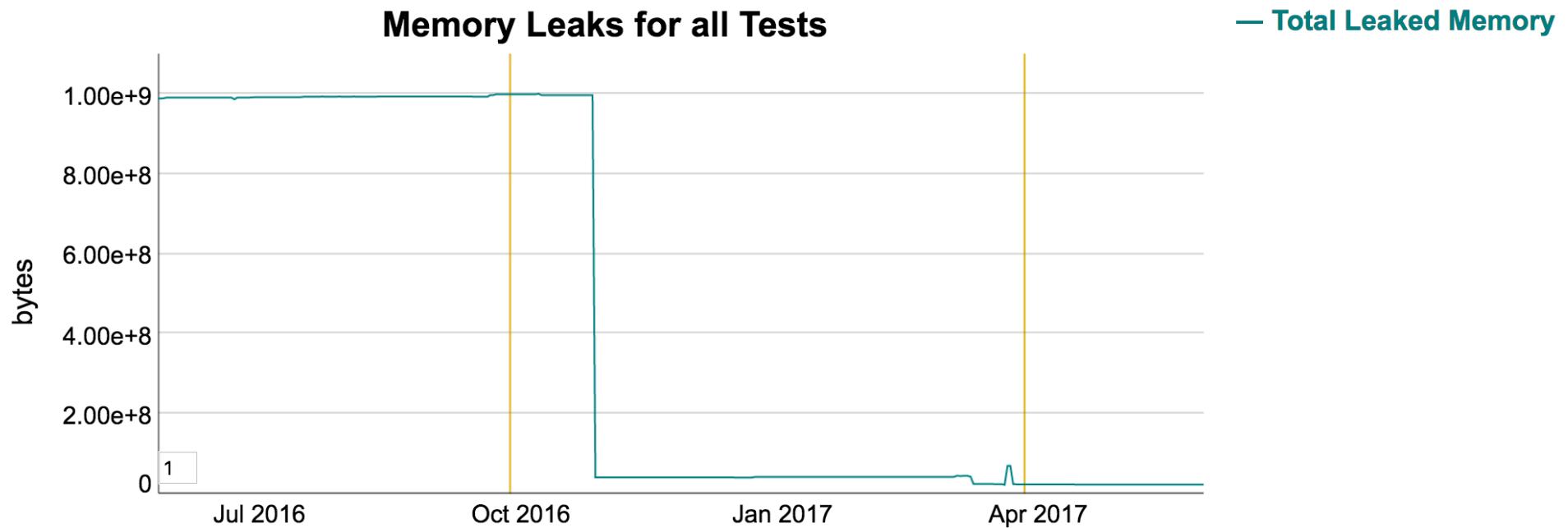
COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

Reduction in Memory Leaks

Summary:

- We've closed the last major source of compiler-introduced leaks:



Next Steps:

- Close user-introduced leaks in tests themselves



COMPUTE

|

STORE

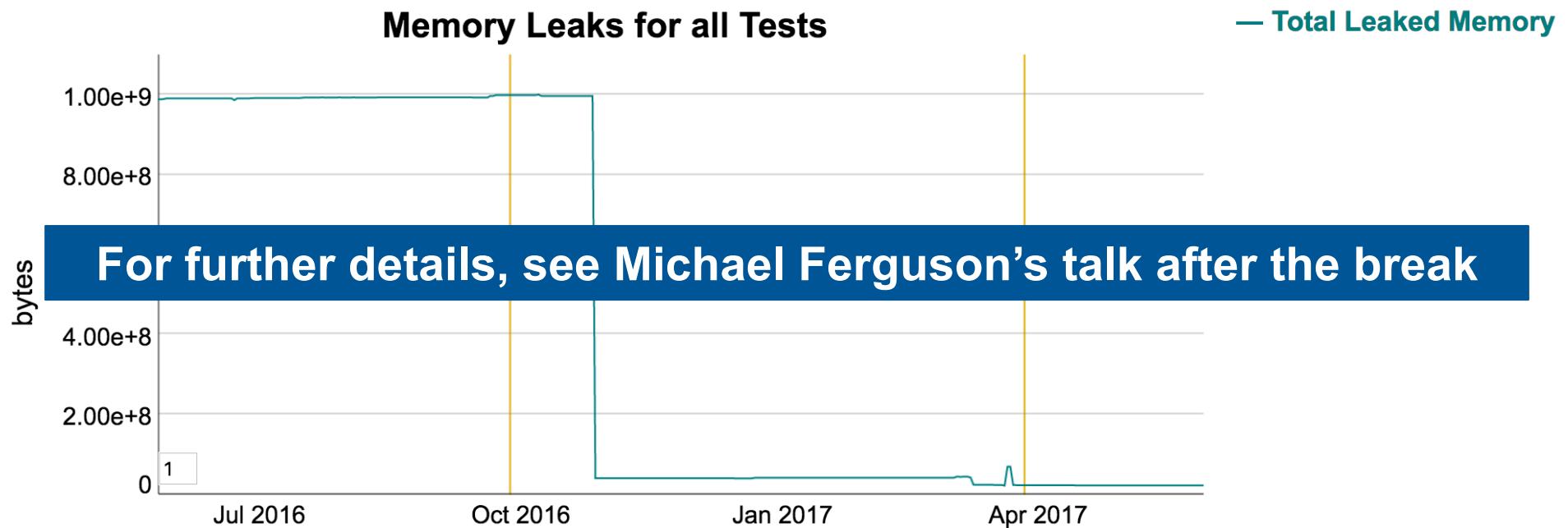
|

ANALYZE

Reduction in Memory Leaks

Summary:

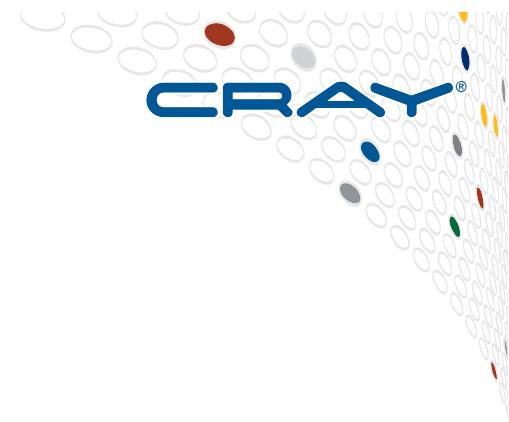
- We've closed the last major source of compiler-introduced leaks:



Next Steps:

- Close user-introduced leaks in tests themselves





Library Improvements



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Library Improvements in 1.14–1.15

New Libraries:

- Date / Time
- Owned / Shared for delete-free class objects
- Futures
- BLAS
- MPI
- ZeroMQ
- BigInteger
- MatrixMarket
- RangeChunk
- LinearAlgebra (first draft)

Improved Libraries:

- FFTW
- Sort/Search



COMPUTE

|

STORE

|

ANALYZE



Library Improvements in 1.14–1.15

New Libraries:

- Date / Time
- Owned / Shared for delete-free class objects
- Futures
- BLAS
- MPI
- ZeroMQ
- BigInteger
- MatrixMarket
- RangeChunk
- LinearAlgebra (first draft)

(contributed by non-Cray developers)

Improved Libraries:

- FFTW
- Sort/Search



Libraries: Summary

[Chapel Documentation 1.15](#)

Search docs

COMPILE AND RUNNING CHAPEL

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

WRITING CHAPEL PROGRAMS

- [Quick Reference](#)
- [Hello World Variants](#)
- [Primers](#)
- [Language Specification](#)
- [Built-in Types and Functions](#)

[Standard Modules](#)

- [Assert](#)
- [Barrier](#)
- [BigInteger](#)
- [BitOps](#)
- [Buffers](#)
- [CommDiagnostics](#)
- [DateTime](#)
- [DynamicIterators](#)
- [FileSystem](#)
- [GMP](#)
- [Help](#)
- [IO](#)
- [List](#)
- [Math](#)
- [Memory](#)
- [Path](#)
- [Random](#)
- [Reflection](#)
- [Regexp](#)
- [Spawn](#)
- [Sys](#)
- [SysBasic](#)
- [SysCTypes](#)
- [SysError](#)
- [Time](#)
- [Types](#)
- [UtilReplicatedVar](#)

Docs » Standard Modules [View page source](#)

Standard Modules

Standard modules are those which define the Standard Library.

All Chapel programs automatically use the following standard modules:

- [Assert](#)
- [Barrier](#)
- [BigInteger](#)
- [BitOps](#)
- [Buffers](#)
- [CommDiagnostics](#)
- [DateTime](#)
- [DynamicIterators](#)
- [FileSystem](#)
- [GMP](#)
- [Help](#)
- [IO](#)
- [List](#)
- [Math](#)
- [Memory](#)
- [Path](#)
- [Random](#)
- [Reflection](#)
- [Regexp](#)
- [Spawn](#)
- [Sys](#)
- [SysBasic](#)
- [SysCTypes](#)
- [SysError](#)
- [Time](#)
- [Types](#)
- [UtilReplicatedVar](#)

[Chapel Documentation 1.15](#)

Search docs

COMPILE AND RUNNING CHAPEL

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

WRITING CHAPEL PROGRAMS

- [Quick Reference](#)
- [Hello World Variants](#)
- [Primers](#)
- [Language Specification](#)
- [Built-in Types and Functions](#)

[Standard Modules](#)

[Package Modules](#)

- [BLAS](#)
- [Curl](#)
- [FFTW](#)
- [FFTW_MT](#)
- [Futures](#)
- [HDFS](#)
- [HDFSSIterator](#)
- [LAPACK](#)
- [LinearAlgebra](#)
- [MPI](#)
- [Norm](#)
- [OwnedObject](#)
- [RangeChunk](#)
- [RecordParser](#)
- [Search](#)
- [SharedObject](#)
- [Sort](#)
- [VisualDebug](#)
- [ZMQ](#)

Docs » Package Modules

Package Modules

Package modules are libraries that currently live outside of the Chapel Standard Library because they are not considered to be fundamental enough or because they are not yet considered good enough for inclusion there.

- [BLAS](#)
- [Curl](#)
- [FFTW](#)
- [FFTW_MT](#)
- [Futures](#)
- [HDFS](#)
- [HDFSSIterator](#)
- [LAPACK](#)
- [LinearAlgebra](#)
- [MPI](#)
- [Norm](#)
- [OwnedObject](#)
- [RangeChunk](#)
- [RecordParser](#)
- [Search](#)
- [SharedObject](#)
- [Sort](#)
- [VisualDebug](#)
- [ZMQ](#)



COMPUTE

STORE

ANALYZE

Libraries: Summary

Docs » Standard Modules View page source

Standard Modules

Standard modules are those which define interfaces to the Standard Library.

All Chapel programs automatically use the Standard Library.

Docs » Package Modules

Package Modules

Summary: Chapel has an increasingly capable suite of libraries

Next Steps: Continue to grow this suite
 Make it simpler for users to do so as well
 Support a Chapel package manager

<ul style="list-style-type: none"> • Asynchronous • Basic • BigData • Build • Collections • Data • Dynamicctors • FileSystem • GMP • Help • IO • List • Math • Memory 	<ul style="list-style-type: none"> • Regexp • Spawn • Sys • SysBasic • SysCTypes • SysError • Time • Types • UtilReplicatedVar 	<ul style="list-style-type: none"> • BLAS • Curl • FFTW • FFTW_MT • Futures • HDFS • HDFSIterator 	<ul style="list-style-type: none"> • OwnedObject • RangeChunk • RecordParser • Search • SharedObject • Sort • VisualDebug • ZMQ
---	---	--	---



COMPUTE

STORE

ANALYZE



Interoperability Improvements

- Users can now pass Chapel function pointers to C
- Extern block support is far more robust

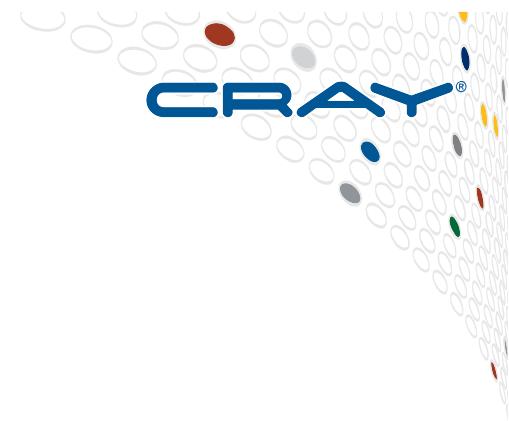
- Sample import of GSL routines from C:

```
extern {
    // Special functions
    #include "gsl/gsl_sf.h"
    // Constants
    #include "gsl/gsl_const.h"
    // Integration
    #include "gsl/gsl_integration.h"
    // Random numbers and distributions
    #include "gsl/gsl_rng.h"
    #include "gsl/gsl_randist.h"
    #include "gsl/gsl_cdf.h"
    // Interpolation
    #include "gsl/gsl_interp.h"
    #include "gsl/gsl_spline.h"
}
```

- Improvements to c2chapel script (see version on master)



COMPUTE | STORE | ANALYZE



Documentation Improvements



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Documentation Improvements

● Significant Expansions / Improvements to online docs:

The screenshot shows three versions of the same documentation page, illustrating the evolution of the content:

- Left Version (Original):** Shows a sidebar with "COMPILING AND RUNNING CHAPEL" and "WRITING CHAPEL PROGRAMS" sections.
- Middle Version (Improved):** Shows expanded sections: "Primers", "Language Basics", "Iterators", "Task Parallelism", "Sync / Singles", and "Atomics".
- Right Version (Final):** Shows the final state with a more detailed "Base Language" section, "Task Parallelism" section, and "Locality" section.

Base Language
This is the core of Chapel and what remains when all features in support of parallelism and locality are removed.

- Hello world: simple console output
- Variable Declarations
- Basic Types: booleans, numbers, and strings
- Literal Values for Basic Types
- Casts: explicit type conversions
- for-loops: structured serial iteration
- Zippered Iteration

(more to come...)

Task Parallelism
These are Chapel's lower-level features for creating parallel explicitly and synchronizing between them.

- Task Parallelism Overview
- begin Statements: unstructured tasking
- cobegin Statements: creating groups of tasks
- coforall-loops: loop-based tasking

(more to come...)

Data Parallelism
These are Chapel's higher-level features for creating parallelism more abstractly using a rich set of data structures.

- forall-loops: data-parallel loops

(more to come...)

Locality
These are Chapel's features for describing how data and tasks should be mapped to the target architecture for the purpose of performance and scalability.



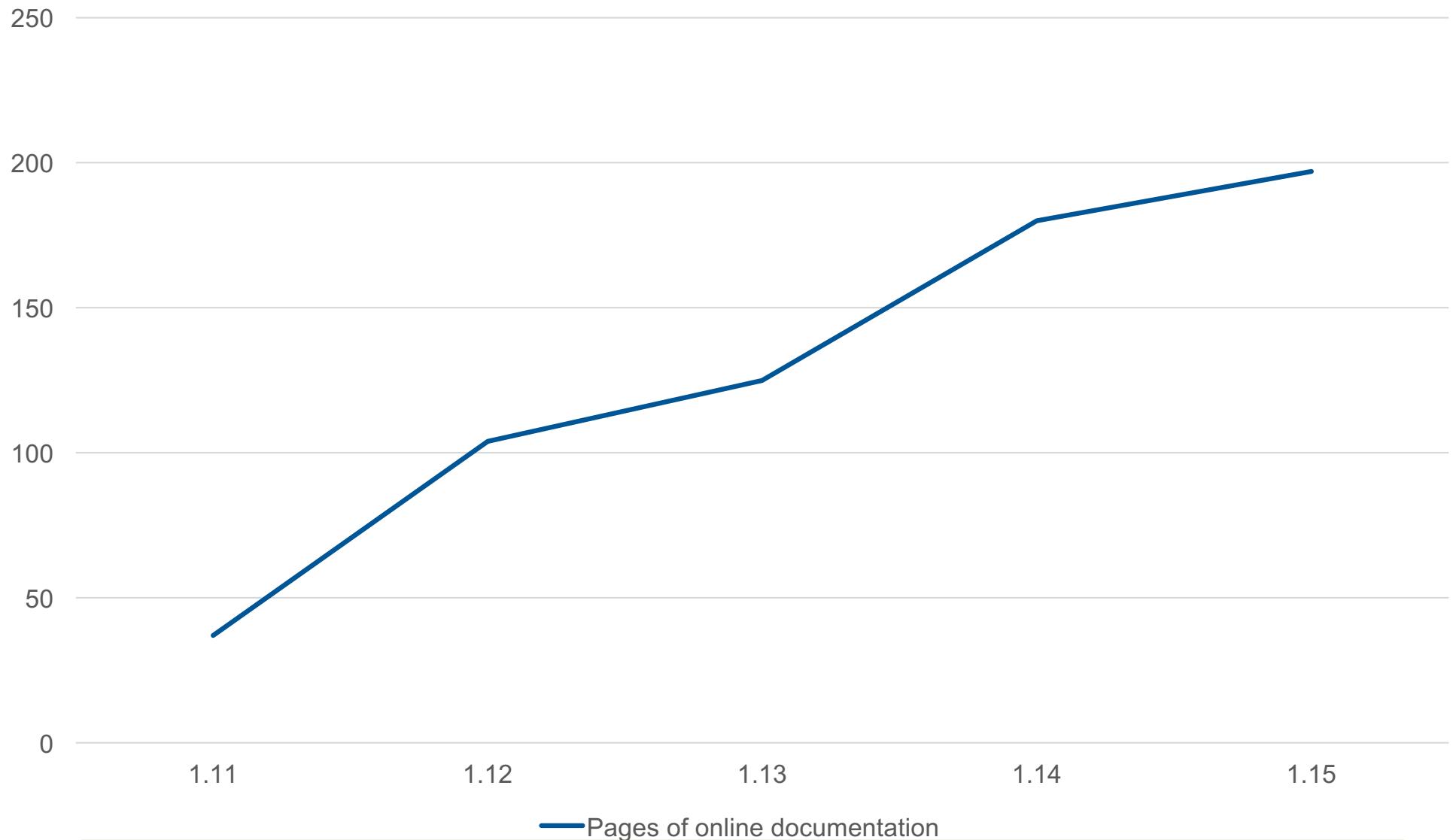
COMPUTE

STORE

ANALYZE

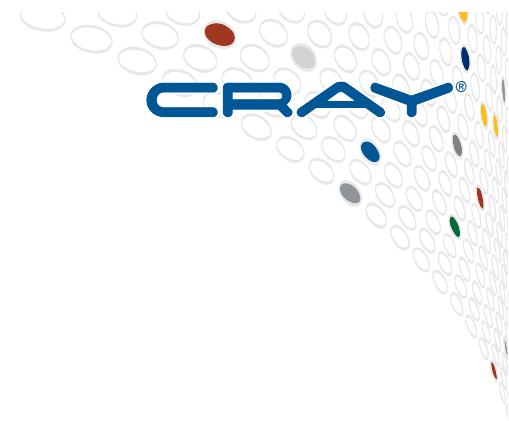


Pages of Online Docs across Releases



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Portability / Packaging Improvements



COMPUTE

|

STORE

|

ANALYZE

Copyright 2017 Cray Inc.

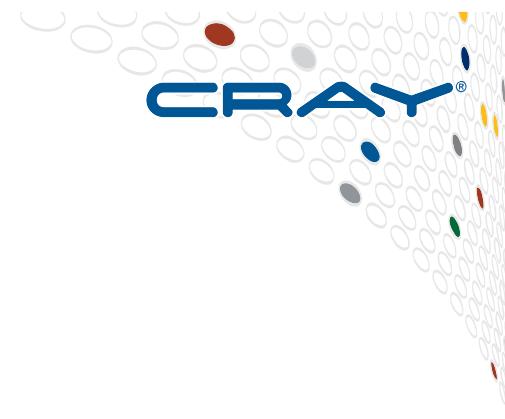


Intel Xeon Phi (“KNL”) locale model

- Chapel can target KNL’s MCDRAM via on-clauses

```
on here.highBandwidthMemory() {  
    x = new myClass();           // placed in MCDRAM  
    ...  
}  
  
on here.defaultMemory() {  
    y = new myClass();          // placed in DDR  
    ...  
}  
  
on y.locale.highBandwidthMemory() {  
    z = new myClass();          // same locale as y, but using MCDRAM  
    ...  
}
```





Other Portability / Packaging Improvements

- AWS EC2
- Windows 10 bash shell
- Docker package now available
- ARM64 support
- Support for Chapel configuration .dotfiles
- Improved portability across various *nix flavors
- **in-progress:**
 - Debian
 - AMD
 - OFI / libfabric



COMPUTE

|

STORE

|

ANALYZE

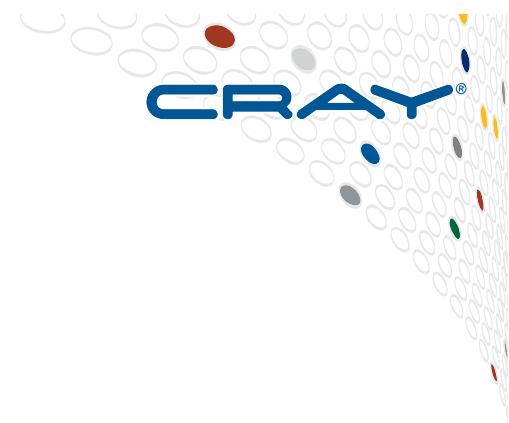


Other Portability / Packaging Improvements

- AWS EC2
- Windows 10 bash shell
- Docker package now available
- ARM64 support
- Support for Chapel configuration .dotfiles
- Improved portability across various *nix flavors
- in-progress:
 - Debian
 - AMD
 - OFI / libfabric

For further details on the AMD and OFI / libfabric efforts, see the portability talks by Mike Chu and Sung-Eun Choi before lunch





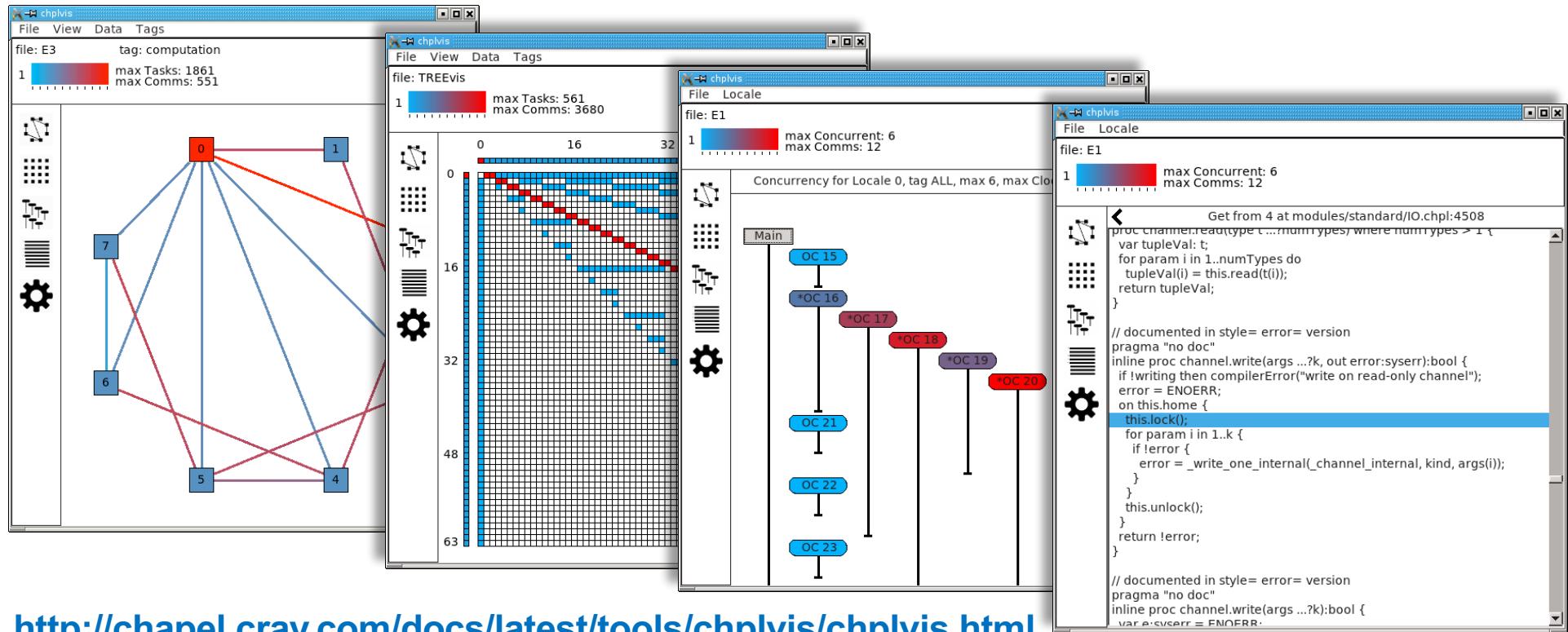
Tool Improvements



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

chplvis: Chapel Execution Visualization Tool



<http://chapel.cray.com/docs/latest/tools/chplvis/chplvis.html>

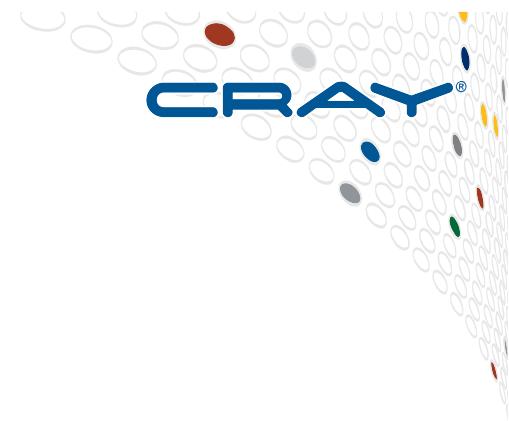


COMPUTE

STORE

ANALYZE

Copyright 2017 Cray Inc.



Core Improvements



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Language Improvements

- **Improvements:**

- improved array semantics
- improved support for generic objects
- first-class ‘void’ type
- module-level de-initializers
- forwarding fields in objects
- fixed where-clause support

- **In-Progress:**

- initializers (constructor replacement)
- error-handling

See Preston Sahabu’s talk this morning



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Domain Map Improvements

New distributions:

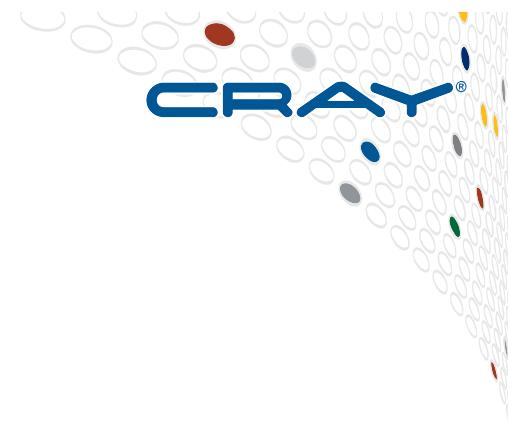
- Stencil distribution
- Sparse Block distribution (needs further tuning)

See early block sparse results in
Ariful Azad's talk this afternoon

Domain map improvements:

- Added locality queries to distributed domains/arrays
- Simplified domain map standard interface





Misc Improvements

- Open-sourced ‘ugni’ communication layer
- Support for stack traces on program halt()s (GSoC project)



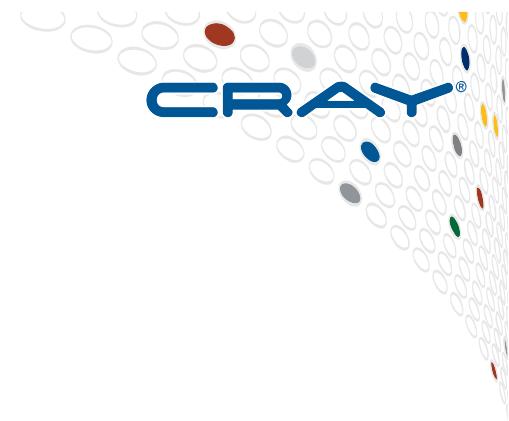
COMPUTE

|

STORE

|

ANALYZE



Meta Stuff



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.

Google Summer of Code 2017

- **Google Summer of Code**
 - Google's way of supporting the open source community
 - Chapel had 2 successful students in GSoC 2016

Google Summer of Code 2017:

- Przemyslaw Lesniak – **LLVM backend**
 - Mentor: Michael Ferguson
- Sarthak Munshi – **Cryptography module**
 - Mentor: Andrea Francesco Iurio
- Louis Jenkins - **Distributed data structures**
 - Mentor: Engin Kayraklıoglu
- Nikhil Mahendran - **Chapel online**
 - Mentors: Ashish Chaudhary and Ben Albrecht



Chapel on Facebook and Twitter

The image displays two social media profiles for the Chapel programming language. On the left is the Facebook page, which features a large green and blue 'C' logo, a post by Jonathan Dursi comparing Chapel and Julia, and a sidebar with navigation links like Home, Posts, Videos, Photos, About, Likes, Promote, and Manage Promotions. On the right is the Twitter profile (@ChapelLanguage), which shows 286 tweets, 13 following, 153 followers, 57 likes, 1 list, and 0 moments. The Twitter feed contains a tweet from May 30th featuring a Chapel program snippet and its execution output.

Chapel Programming Language

As though it weren't enough that we asked him to give a keynote talk for us at CHI UW 2017 (at IPDPS) this week, Jonathan Dursi has written a new blog post comparing and evaluating Chapel and Julia:

<https://www.dursi.ca/post/julia-vs-chapel.html>

Should I use Chapel or Julia for my next project?
R&D computing at scale.
DURSI.CA

188 people reached

Boost Post

Like Comment Share

Vladimír Fuka, Russel Winder and 3 others

1 share

Write a comment...

Chapel Language @ChapelLanguage · May 30
If you've just joined us, we've been tweeting complete 140-character Chapel programs for fun (#Chapel140). Here's a silly "O(n)" sort entry:

```
A.push_back(a);

coforall a in A {
    sleep(a);
    writeln(a);
}

$ cat sleepsrt.stdin
5 2 9 6 8 1 4 10 3 7
$ ./sleepsort < sleepsrt.stdin
1
2
3
4
5
6
```



COMPUTE

STORE

ANALYZE



Chapel YouTube Channel

YouTube Search

Chapel Parallel Programming Language 29 subscribers

HOME VIDEOS PLAYLISTS CHANNELS ABOUT

Chapel videos PLAY ALL

The Audacity of Chapel: Scalable Parallel Programming Done Right - Brad Chamberlain [ACCU 2017]
ACCU Conference • 99 views • 1 day ago
Programming language designers have to date largely failed the large-scale parallel computing community, and arguably even parallel programmers targeting desktops or modest-scale clusters.
53:54

CHIUW 2016 keynote: "Chapel in the (Cosmological) Wild", Nikhil Padmanabhan
Chapel Parallel Programming Language • 279 views • 10 months ago
This is Nikhil Padmanabhan's keynote talk from CHIUW 2016: the 3rd Annual Chapel Implementers and Users workshop. The slides are available at: <http://chapel.cray.com/CHIUW/2016/Padmanabhan->
56:14

Chapel Productive, Multiresolution Parallel Programming | Brad Chamberlain, Cray, Inc.
ANL Training • 671 views • 7 months ago
Presented at the Argonne Training Program on Extreme-Scale Computing, Summer 2016. Slides for this presentation are available here: <http://extremecomputingtraining.anl.gov/sessions/chapel-producti...>
56:47

Pycon 2016: Fast Python! Don't Bother?
PyCon UK • 4.5K views • 7 months ago



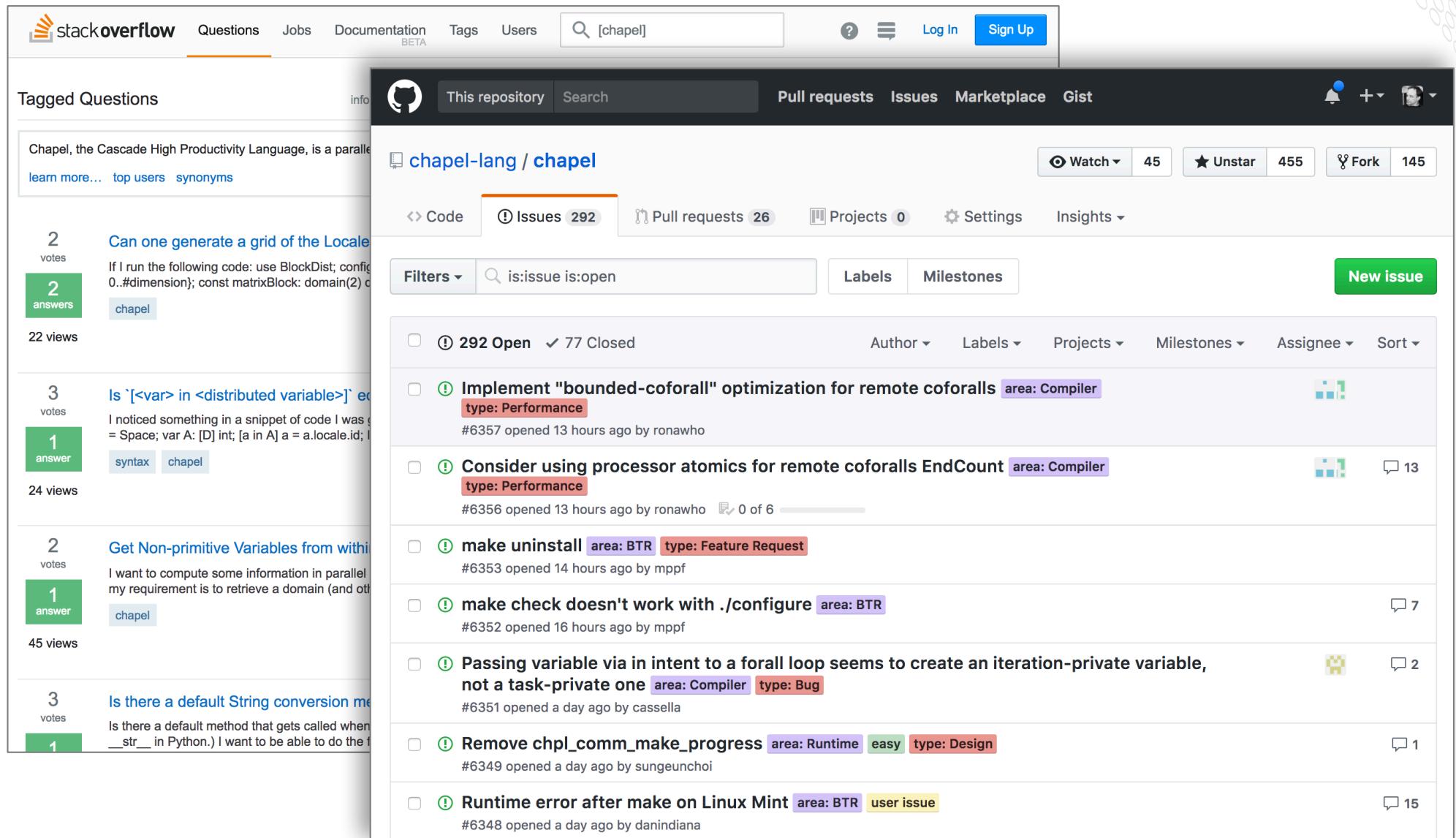
COMPUTE

STORE

ANALYZE

Copyright 2017 Cray Inc.

Chapel StackOverflow and GitHub Issues



The screenshot shows two side-by-side web pages related to the Chapel programming language.

Stack Overflow (Left):

- Tagged Questions:** Shows a list of questions tagged with "chapel".
- Can one generate a grid of the Locale?** (2 votes, 2 answers, 22 views)

If I run the following code; use BlockDist; config 0..#dimension); const matrixBlock: domain(2) {
- Is '<var> in <distributed variable>' ever valid?** (3 votes, 1 answer, 24 views)

I noticed something in a snippet of code I was looking at. It had this: var A: [D] int; [a in A] a = Space; var A: [D] int; [a in A] a = a.locale.id; I'm not sure if this is valid or not.
- Get Non-primitive Variables from within a Domain** (2 votes, 1 answer, 45 views)

I want to compute some information in parallel over a domain. My requirement is to retrieve a domain (and other information) from within a domain. I am not sure how to do this.
- Is there a default String conversion method?** (3 votes, 1 answer, 1 view)

Is there a default method that gets called when I do `str` in Python.) I want to be able to do the same thing in Chapel.

Github Issues (Right):

- chapel-lang / chapel** repository (Watch 45, Unstar 455, Fork 145)

This repository contains 292 open issues and 26 pull requests.
- Issues Tab:** Filtered to show open issues. The first few are:
 - Implement "bounded-coforall" optimization for remote coforalls (area: Compiler, type: Performance)
 - Consider using processor atomics for remote coforalls EndCount (area: Compiler, type: Performance)
 - make uninstall (area: BTR, type: Feature Request)
 - make check doesn't work with ./configure (area: BTR)
 - Passing variable via intent to a forall loop seems to create an iteration-private variable, not a task-private one (area: Compiler, type: Bug)
 - Remove chpl_comm_make_progress (area: Runtime, easy, type: Design)
 - Runtime error after make on Linux Mint (area: BTR, user issue)



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Chapel on cyber-dojo

cyber-dojo.org
the place to practice programming



[setup a new practice session](#)
[enter a practice session](#)
[review a practice session](#)

100% of your donation buys Raspberry Pi computers to help children learn to program [please donate](#)

cyber-dojo Foundation

Commercial use of the public server requires a license
The cyberdojo Foundation issues licenses
100% of the license fees buy Raspberry Pi computers to help children learn to program
Hosting fees for the public server are paid by Cucumber Limited

Coimbatore, India 

Bray, Ireland 

Scottish Charitable Incorporated Organisation (magic number SC045890)

about

cyber-dojo is open sourced on [github](#)
Nadya Sivers drew the fantastic animal images
Jon Jagger designs and builds cyber-dojo



setup a new practice session

[switch to custom choices](#)

language?

The starting files for your chosen language+tests are always a function called `answer` that returns `6 * 9` and a test called `life, the universe, and everything that expects 42`

The starting files are *unrelated* to your chosen exercise. They are simply an example to start you off.

Tests must complete in 10 seconds.

tests?

assert

Asm
BCPL
Bash
C (clang)
C (gcc)
C#
C++ (clang++)
C++ (g++)
Chapel
Clojure
CoffeeScript
D
Elixir
Elm
Erlang
F#
Fortran

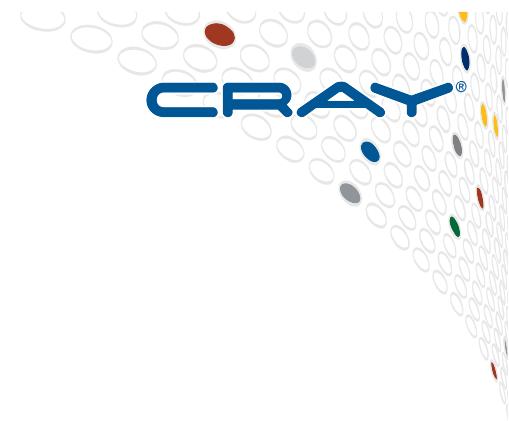
next



COMPUTE

STORE

ANALYZE



What's Next?



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Top Chapel Priorities for version 1.16

- **Wrap up key language features:**
 - initializers
 - error-handling
 - delete-free class idioms (Shared, Owned)
- **Package Manager**
- **Benchmark- / App- / User-driven...**
 - ...performance tuning
 - ...library expansions



COMPUTE

|

STORE

|

ANALYZE



Our #1 Challenge

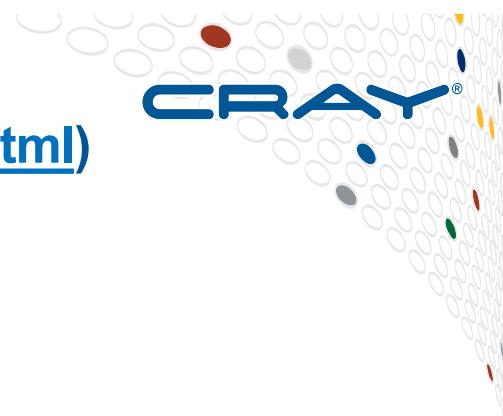
- **How to grow the user and developer communities?**
- **How to encourage people to look at Chapel again?**
 - overcome impressions made in our young, awkward years...

'Scientific computing communities are very wary of new technologies (it took 10+ years for Python to start getting any traction), with the usual, self-fulfilling, fear being “what if it goes away?”'

- Jonathan Dursi, from *Should I Use Chapel or Julia for my next project?*



COMPUTE | STORE | ANALYZE



CHIUIW 2017: Agenda (chapel.cray.com/CHIUIW2017.html)

- 8:30: Chapel Boot Camp (optional)
- 9:00: **Welcome, State of the Project**
- 9:30: **Break**
- 10:00: **Talks: Chapel Design and Implementation**
- 11:10: **Quick Break**
- 11:20: **Talks: Targeting New Architectures**
- 12:00: **Lunch**
- 1:30: **Keynote Talk: Jonathan Dursi**
- 2:30: **Talks: Uses of Chapel**
- 3:20: **Break**
- 3:50: **Talks: Benchmarking and Performance**
- 4:40: **Lightning Talks and Flash Discussions**
- 5:30: **Wrap-up / Head to Dinner**



COMPUTE | STORE | ANALYZE

Copyright 2017 Cray Inc.



Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

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

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, 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