



**Hewlett Packard  
Enterprise**

# **CHIUW 2023: STATE OF THE CHAPEL PROJECT**

---

Brad Chamberlain  
June 2, 2023

# WELCOME TO THE 10<sup>TH</sup> ANNUAL CHIUW WORKSHOP!



Can a single parallel language be ...  
... as productive as Python?  
... as fast as Fortran?  
... as scalable as MPI?

*Attend CHIUW on Friday and hear about how the Chapel community is working to make this vision a reality!*

<b>Introduction to Chapel, State of the Project</b>
8:30 - 9:00: Brad Chamberlain, Cray Inc.
<b>Technical Talks: Application Studies</b>
9:00 - 9:30: <a href="#">User Experiences with a Chapel Implementation of UTS</a> Claudia Fohry (Universität Kassel, Germany), Jens Breitbart (Technische Universität München, Germany)
9:30 - 10:00: <a href="#">Evaluating Next Generation PGAS Languages for Computational Chemistry</a> Daniel Chavarria-Miranda, Sriram Krishnamoorthy, Joseph Manzano, Abhirav Vishnu (Pacific Northwest National Laboratory)
10:00 - 10:20: Break
<b>Technical Talks: Language Extensions</b>
10:30 - 11:00: <a href="#">Programmer-Guided Reliability in Chapel</a> David E. Bernholdt, Wael R. Elwasif, Christos Kartsakis, Seyong Lee, Tiffany M. Mintz (Oak Ridge National Laboratory)
11:00 - 11:30: <a href="#">Towards Interfaces for Chapel</a> Chris Waller, Jeremy G. Siek (Indiana University)
<b>Technical Talks: Compiler Optimizations</b>
11:30 - 12:00: <a href="#">Affine Loop Optimization using Module Unlinking in Chapel</a> Asoon Sharifi, Rajeev Barua (University of Maryland), Michael Ferguson (Laboratory for Telecommunication Sciences)
12:00 - 1:00: Lunch (on your own)
<b>Invited Talk: Robert Harrison</b>
1:00 - 1:45: <a href="#">Walking to the Chapel</a> Robert Harrison (Princeton University/Brownhaven National Laboratory)
<b>Abstract:</b> MADNESS is a general-purpose numerical environment that sits upon a scalable runtime that consciously includes elements "borrowed" from other projects including Charm++ and the HPCS programming languages, including Chapel; it is also designed to be interoperable with "legacy" code. But as I have said to Chapel's architects several times, maintaining our own runtime is just an unpleasant transitional phase and we are looking for a permanent home—is that home Chapel? I'll give you some flavor of what MADNESS does and how, with the objective of starting a conversation and seedling collaborations.
<b>Technical Talks: Compiler Optimizations (continued)</b>
1:45 - 2:15: <a href="#">LLVM Optimizations for PGAS Programs</a> Abhinav Hazra, Rajeev Barua, Jieheng Zhao, Vivek Sarkar (Rutgers University), Michael Ferguson (Laboratory for Telecommunication Sciences)
<b>Technical Talks: Runtime Improvements</b>
2:15 - 2:45: <a href="#">Opportunities for Integrating Tasking and Communication Layers</a> Dylan T. Stark, Brian W. Barrett (Sandia National Laboratories)
2:45 - 3:15: <a href="#">Caching in on Aggregation</a> Michael Ferguson (Laboratory for Telecommunication Sciences)
3:15 - 3:30: Break
<b>Community/Panel Discussion</b>
3:30 - 4:30: Anyone who is interested

## CHIUW: Chapel Implementers and Users Workshop

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

Friday, May 23rd, 2014  
Advance Program

The Chapel Implementers and Users Workshop, to be held in conjunction with [IPDPS 2014](#), will be the first in what is anticipated to be an annual series of workshops designed to bring developers and users of the Chapel language (<http://chapel.cray.com>) together to report on work being done with the language across the broad open-source community. Attendance is open to anyone interested in Chapel, from the most seasoned Chapel user or developer to someone simply curious to learn more. On behalf of the Chapel community and CHIUW steering committee, we hope to see you at CHIUW!

# WELCOME TO THE 10<sup>TH</sup> ANNUAL CHIUW WORKSHOP!



CHIUW 2

Can a single parallel  
... as productive as  
... as fast as Fortran  
... as scalable as MPI?

Attend CHIUW on Friday and hear  
what our community is working to make

#### Introduction to Chapel, State of the Project

8:30 - 9:00: Brad Chamberlain, Cray Inc.

#### Technical Talks: Application Studies

9:00 - 9:30: User Experiences with a Chapel Implementation of UTS

Claudia Fohry (Universität Kassel, Germany), Jens Breitbarth (Technische Universität Münster, Germany)

9:30 - 10:00: Evaluating Next Generation PGAS Languages for Computational Chemistry

Daniel Chavarria-Miranda, Sriram Krishnamoorthy, Joseph Manzano, Abhiram Vishnu

10:00 - 10:20: Break

#### Technical Talks: Language Extensions

10:30 - 11:00: Programmer-Guided Reliability in Chapel

Dougal D. Bernholdt, Wael R. Elwasif, Christos Kartsakis, Seyong Lee, Tiffany M. Mintz

11:00 - 11:30: Towards Interfaces for Chapel

Chris Waller, Jeremy G. Siek (Indiana University)

#### Technical Talks: Compiler Optimizations

11:30 - 12:00: Affine Loop Optimization using Module Unrolling in Chapel

Aaron Sharpe, Rajeev Barua (University of Maryland), Michael Ferguson (Laboratory for Telecommunications Sciences)

12:00 - 1:00: Lunch (on your own)

#### Invited Talk: Robert Harrison

1:00 - 1:45: Walking to the Chapel

1:45 - 2:30: MADNESS: A general-purpose numerical environment that builds upon a set of parallel numerical libraries and tools, including Chapel, that have been developed at Sandia National Laboratories

Abstract: MADNESS is a general-purpose numerical environment that builds upon a set of parallel numerical libraries and tools, including Chapel, that have been developed at Sandia National Laboratories

said to Chapel's architects several times, maintaining our own runtime is just an unkindness to Chapel's users. I'll give you some flavor of what MADNESS does and how, with the help of Chapel, it does it.

#### Technical Talks: Compiler Optimizations (continued)

1:45 - 2:15: LLVM Optimizations for PGAS Programs

Abhishek Rayamajhi, Ross Agius, Jieheng Zhao, Vivek Sarkar (Rice University), Michael Ferguson (Laboratory for Telecommunications Sciences)

#### Technical Talks: Runtime Improvements

2:15 - 2:45: Opportunities for Integrating Tasking and Communication Layers

Dogan T. Suciu, Brian W. Barrett (Sandia National Laboratories)

2:45 - 3:15: Caching in on Application Performance

Michael Ferguson (Laboratory for Telecommunications Sciences)

3:15 - 3:30: Break

#### Community/Panel Discussion

3:30 - 4:30: Anyone who is interested



## Workshop

with [IPDPS 2014](#), will be a series of workshops designed to bring together researchers and practitioners to report on work in progress. Attendance is open to anyone who is interested in contributing to someone's simply

On this day

9 years ago



Brad Chamberlain is with Lydia Duncan.

May 23, 2014 · 1 person

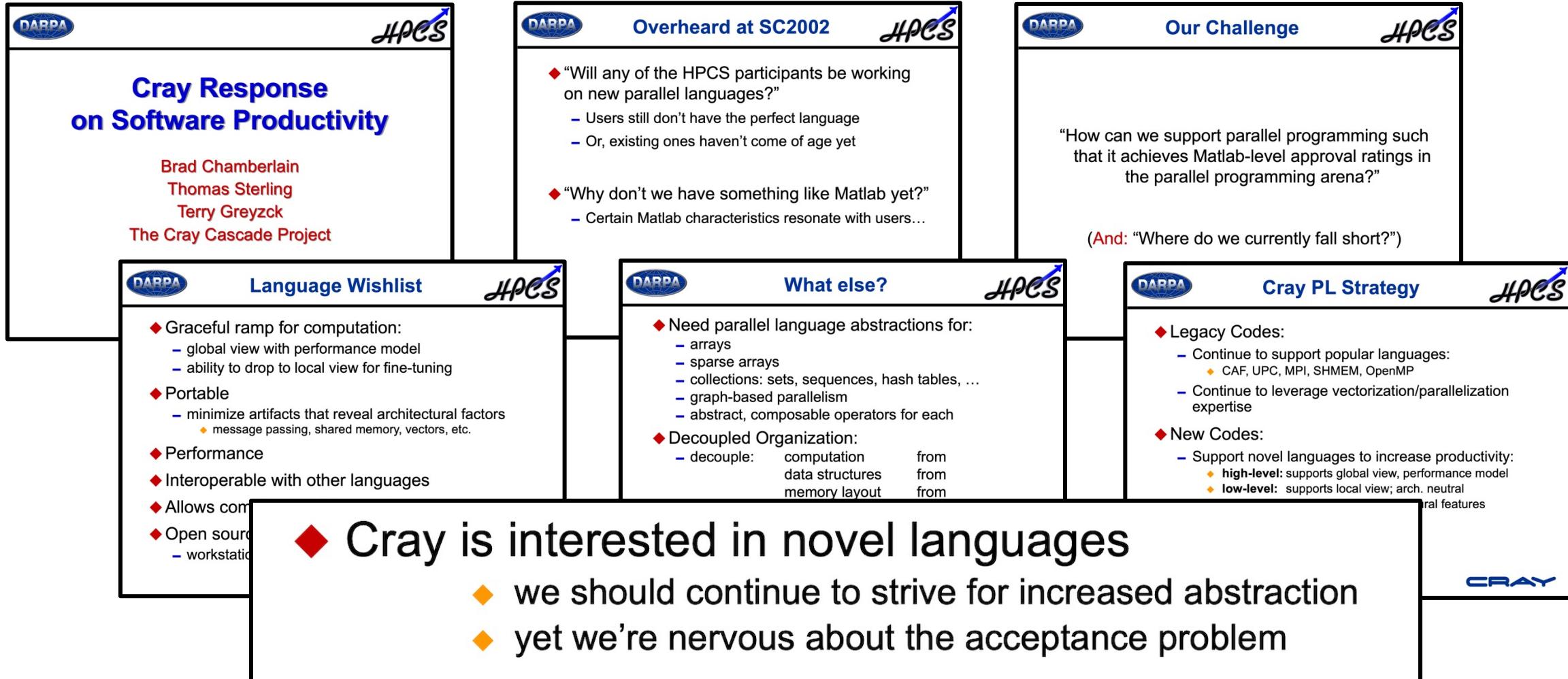
...

still had a solid crowd at the end of a workshop on Chapel at the end of a week-long conference. Not a bad way to kick off what we intend to be an annual event.

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

# CHAPEL'S TURNING 20?!?

Cray first expressed its intention of developing new language(s) as part of HPCS in January 2003



(slides excerpted from a presentation at HPCS Software Productivity Workshop, January 16, 2003)

# WHAT IS CHAPEL?

---

**Chapel:** A modern parallel programming language

- portable & scalable
- open-source & collaborative



## Goals:

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



# HPC BENCHMARKS: CONVENTIONAL APPROACHES VS. CHAPEL

## STREAM TRIAD: C + MPI + OPENMP

```

/*include <hpcc.h>
#ifndef _OPENMP
#include <omp.h>
#endif

static int VectorSize;
static double *a, *b, *c;

int HPCC_StarStream(HPCC_Params *params) {
    int myRank, commSize;
    int rv, errCount;
    MPI_Comm comm = MPI_COMM_WORLD;
    MPI_Comm_size(comm, &commSize);
    MPI_Comm_rank(comm, &myRank);

    rv = HPCC_Stream(params, 0 == myRank);
    MPI_Reduce(&rv, &errCount, 1, MPI_INT, MPI_SUM, 0, comm);

    return errCount;
}

int HPCC_Stream(HPCC_Params *params, int doIO) {
    register int j;
    double scalar;
    VectorSize = HPCC_LocalVectorSize( params, 3, sizeof(double), 0 );
    a = HPCC_XMALLOC( double, VectorSize );
    b = HPCC_XMALLOC( double, VectorSize );
    c = HPCC_XMALLOC( double, VectorSize );

    if (doIO) {
        #ifdef _OPENMP
        #pragma omp parallel
        #endif
        for (j=0; j < VectorSize; j++)
            a[j] = b[j];
        scalar = 3.0;
        #ifdef _OPENMP
        #pragma omp parallel
        #endif
        for (j=0; j < VectorSize; j++)
            b[j] = a[j] + scalar;
    }

    A = B + alpha * C;
}

```

use **BlockDist**;

```

config const n = 1_000_000,
      alpha = 0.01;

const Dom = Block.createDomain({1..n});

var A, B, C: [Dom] real;

B = 2.0;
C = 1.0;

A = B + alpha * C;

```

## HPCC RA: MPI KERNEL

```

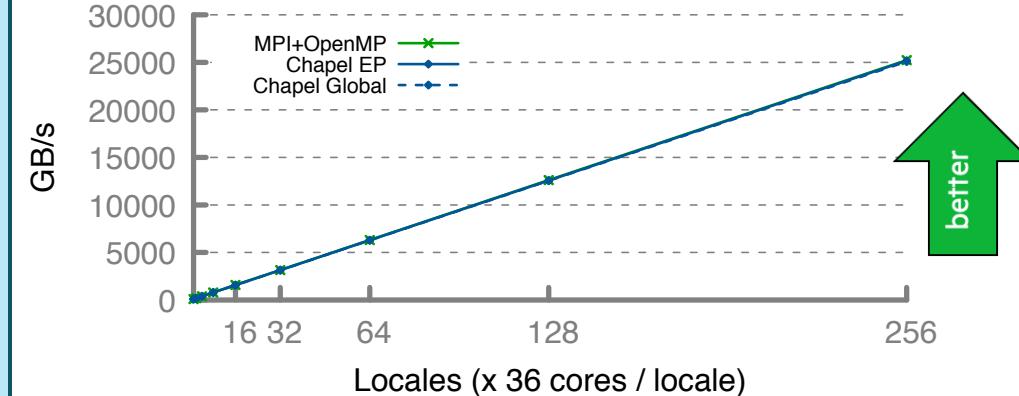
/* Perform updates to main table. The scalar equivalent is:
 * for (i=0;i<RASize;i++) {
 *     Ra[i] = Ra[i] - 27.0*Ra[i]*Ra[i] + 0.7*POLY[i];
 * }
 * TableOffset = (RASize-1)*Ra;
 * ...
 */

MPI_Irecv((LocalRaBuffer, localBufferSize, tparams.dtyped4,
           MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, iinseq);
while (i < SendCount) {
    /* receive message */
    MPI_DemandMsg(&haveDone, &status);
    if (status.MPI_TAG == UPDATE_TAG) {
        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
            if (status.MPI_TAG == UPDATE_TAG) {
                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                            if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                            if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                            if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                            if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                        if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                            if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                                if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
                                                                                                    if (status.MPI_SOURCE == MPI_SELF_SOURCE) {
................................................................
forall (_, r) in zip(Updates, RAStream()) do
    T[r & indexMask].xor(r);
...

```

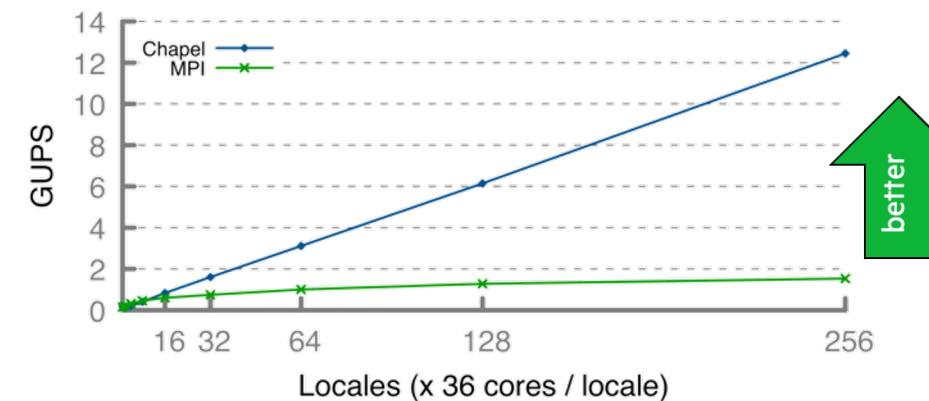
| 72

STREAM Performance (GB/s)



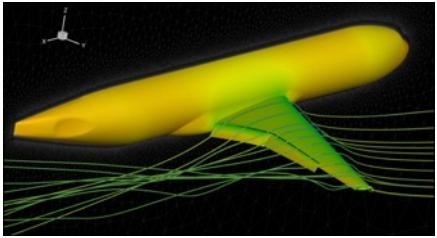
better

RA Performance (GUPS)



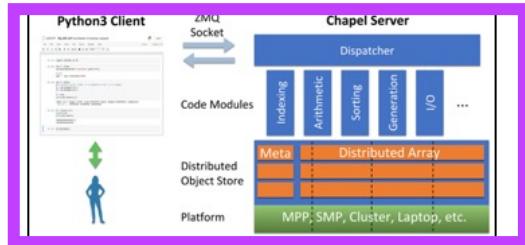
better

# APPLICATIONS OF CHAPEL



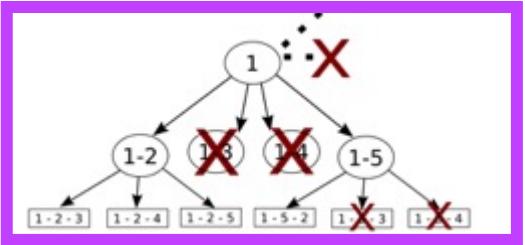
**CHAMPS: 3D Unstructured CFD**

Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.  
École Polytechnique Montréal



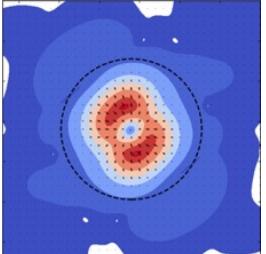
**Arkouda: Interactive Data Science at Massive Scale**

Mike Merrill, Bill Reus, et al.  
U.S. DoD



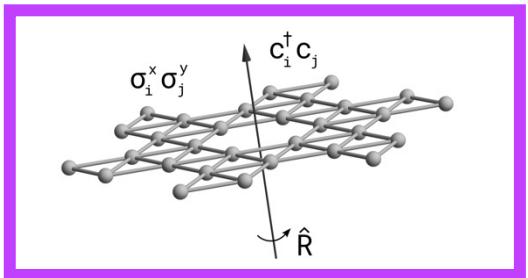
**ChOp: Chapel-based Optimization**

T. Carneiro, G. Helbecque, N. Melab, et al.  
INRIA, IMEC, et al.



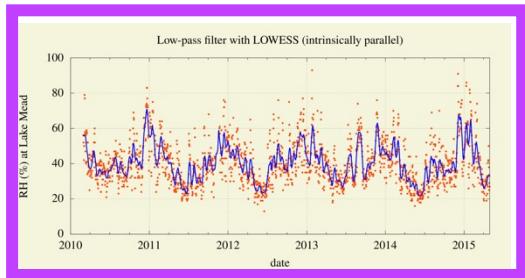
**ChplUltra: Simulating Ultralight Dark Matter**

Nikhil Padmanabhan, J. Luna Zagorac, et al.  
Yale University et al.



**Lattice-Symmetries: a Quantum Many-Body Toolbox**

Tom Westerhout  
Radboud University



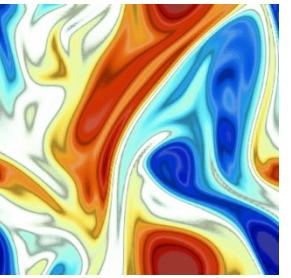
**Desk dot chpl: Utilities for Environmental Eng.**

Nelson Luis Dias  
The Federal University of Paraná, Brazil



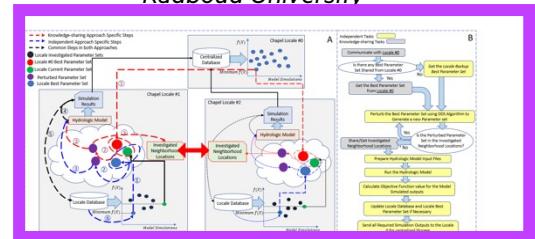
**RapidQ: Mapping Coral Biodiversity**

Rebecca Green, Helen Fox, Scott Bachman, et al.  
The Coral Reef Alliance



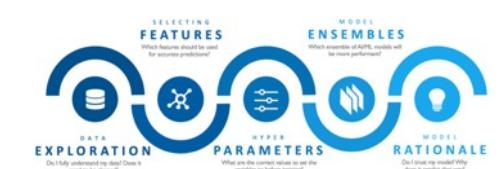
**ChapQG: Layered Quasigeostrophic CFD**

Ian Grooms and Scott Bachman  
University of Colorado, Boulder et al.

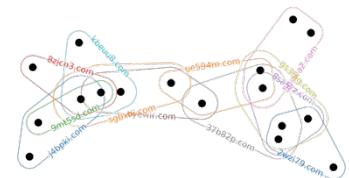


**Chapel-based Hydrological Model Calibration**

Marjan Asgari et al.  
University of Guelph



**CrayAI HyperParameter Optimization (HPO)**



**CHGL: Chapel Hypergraph Library**



**Your Application Here?**

Much more about Applications of Chapel throughout the day

(Images provided by their respective teams and used with permission)

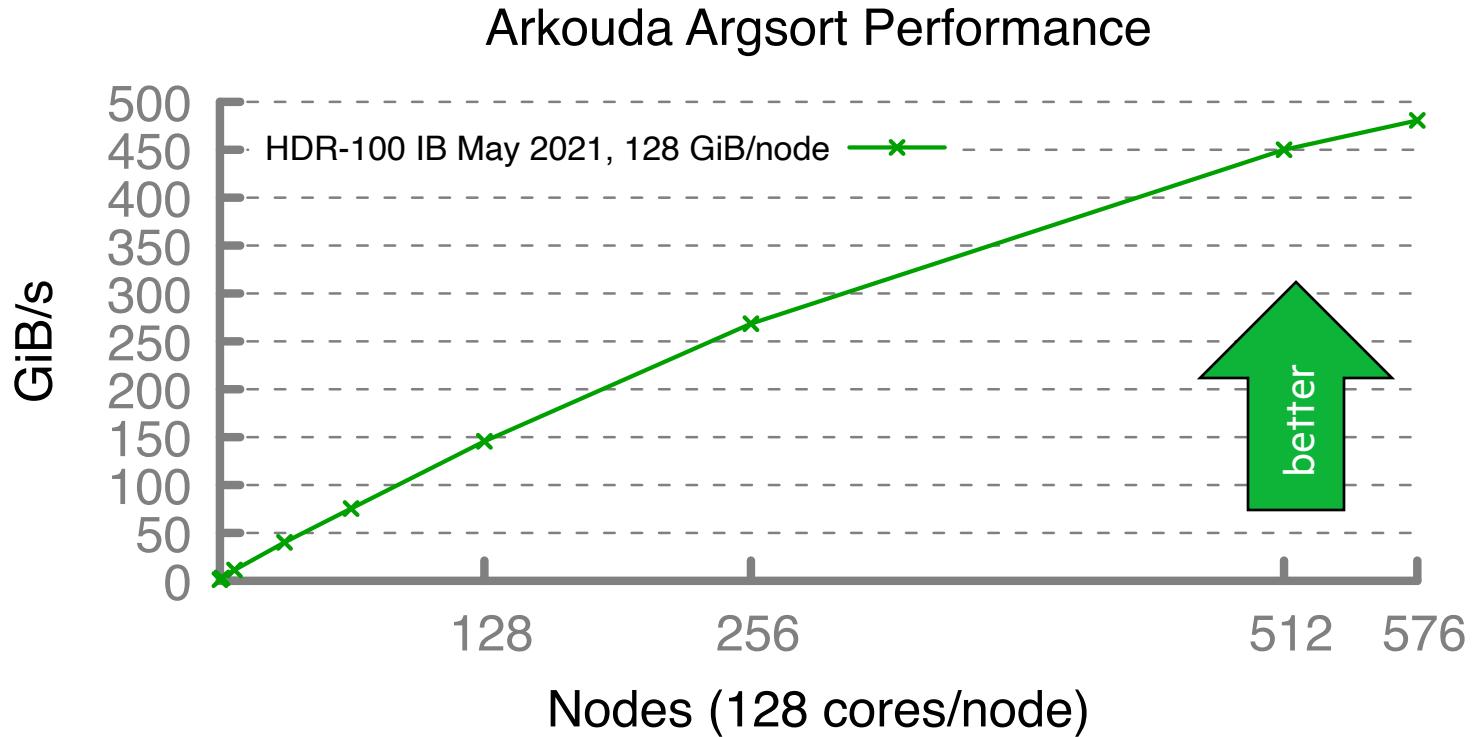
# **CHAPEL ON HPE CRAY EX / SLINGSHOT-11**



# ARKOUDA ARG SORT PERFORMANCE: CHIUW 2022

## HPE Apollo (May 2021)

- HDR-100 Infiniband network (100 Gb/s)
- 73,728 cores / 576 nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)



A notable performance achievement in ~100 lines of Chapel



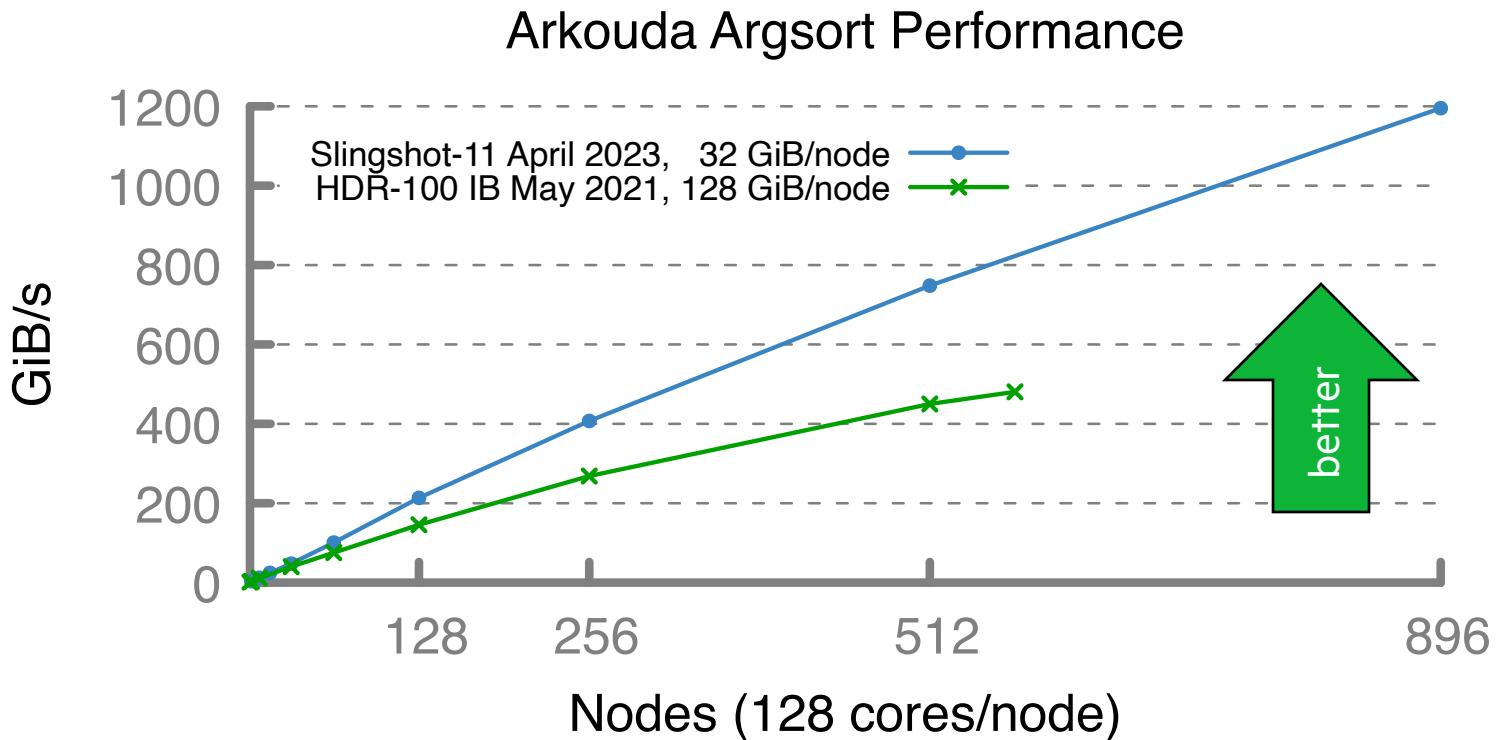
# ARKOUDA ARG SORT PERFORMANCE: TODAY

## HPE Apollo (May 2021)

- HDR-100 Infiniband network (100 Gb/s)
- 73,728 cores / 576 nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)

## HPE Cray EX (April 2023)

- Slingshot-11 network (200 Gb/s)
- 114,688 cores / 896 nodes
- 28 TiB of 8-byte values
- ~1200 GiB/s (~24 seconds)



A notable performance achievement in ~100 lines of Chapel



# ARKOUDA ARG SORT PERFORMANCE: TODAY

## HPE Apollo (May 2021)

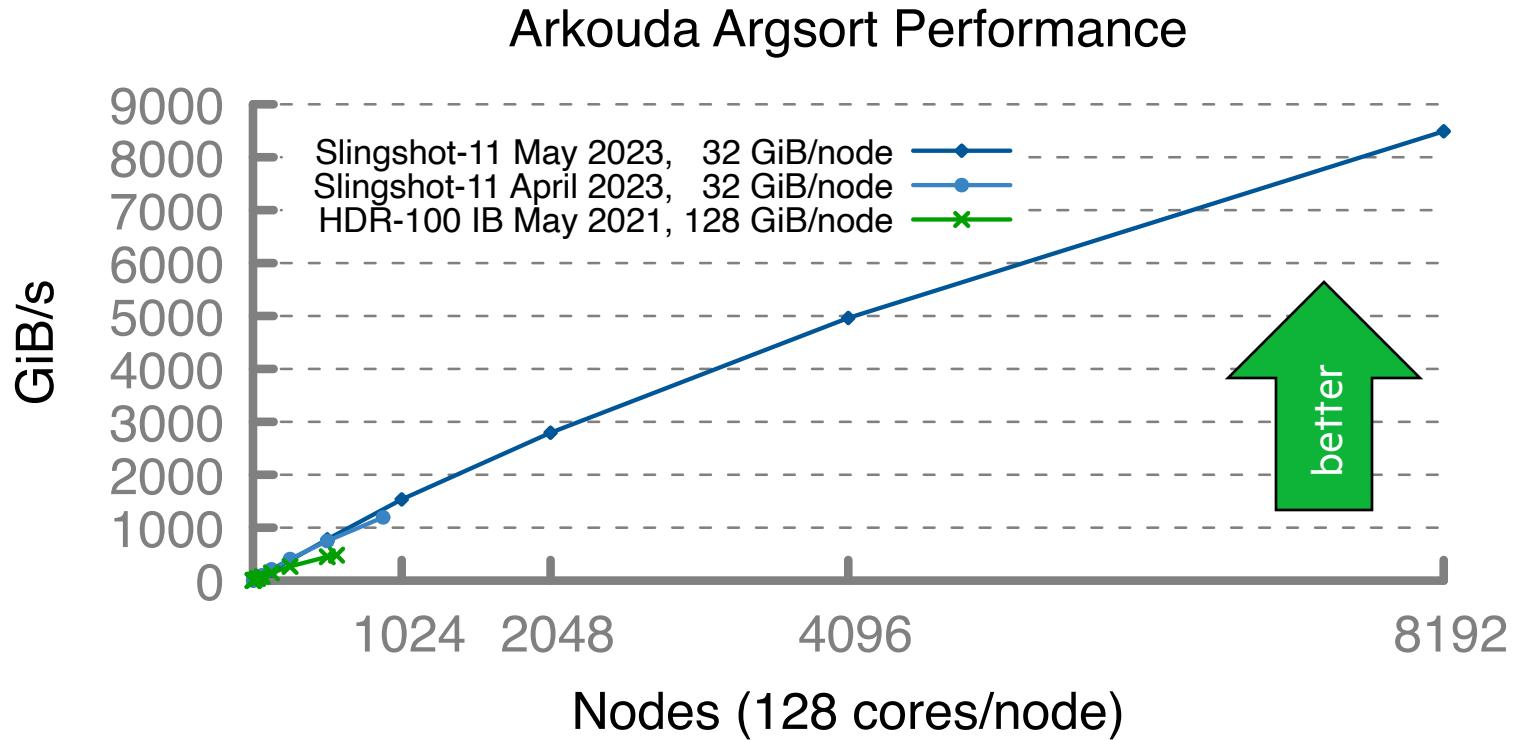
- HDR-100 Infiniband network (100 Gb/s)
- 73,728 cores / 576 nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)

## HPE Cray EX (April 2023)

- Slingshot-11 network (200 Gb/s)
- 114,688 cores / 896 nodes
- 28 TiB of 8-byte values
- ~1200 GiB/s (~24 seconds)

## HPE Cray EX (May 2023)

- Slingshot-11 network (200 Gb/s)
- 1,048,576 cores / 8192 nodes
- 256 TiB of 8-byte values
- ~8500 GiB/s (~31 seconds)



A notable performance achievement in ~100 lines of Chapel



# HPE CRAY EX IMPROVEMENTS

---

## Other New Features on HPE Cray EX:

- ability to run multiple locales per compute node
  - locale per NIC
  - locale per socket
- ability to devote a core to handling communication

## What's Next?

- Extend the above features to other platforms
  - most notably GASNet over InfiniBand
- Perform additional benchmarking studies at scale (HPCC, Bale, PRK, ...)
  - comparing to reference versions in MPI and SHMEM



# **DYNO: REVAMPING THE CHAPEL COMPILER**



# DYNO IN A NUTSHELL

---

## Motivation:

- The Chapel compiler was originally written quickly, by a small team, as a research project
- As a result, it tends to be...
  - ...slow
  - ...difficult to understand when there are errors
  - ...not terribly well-architected: inflexible, challenging to get started with

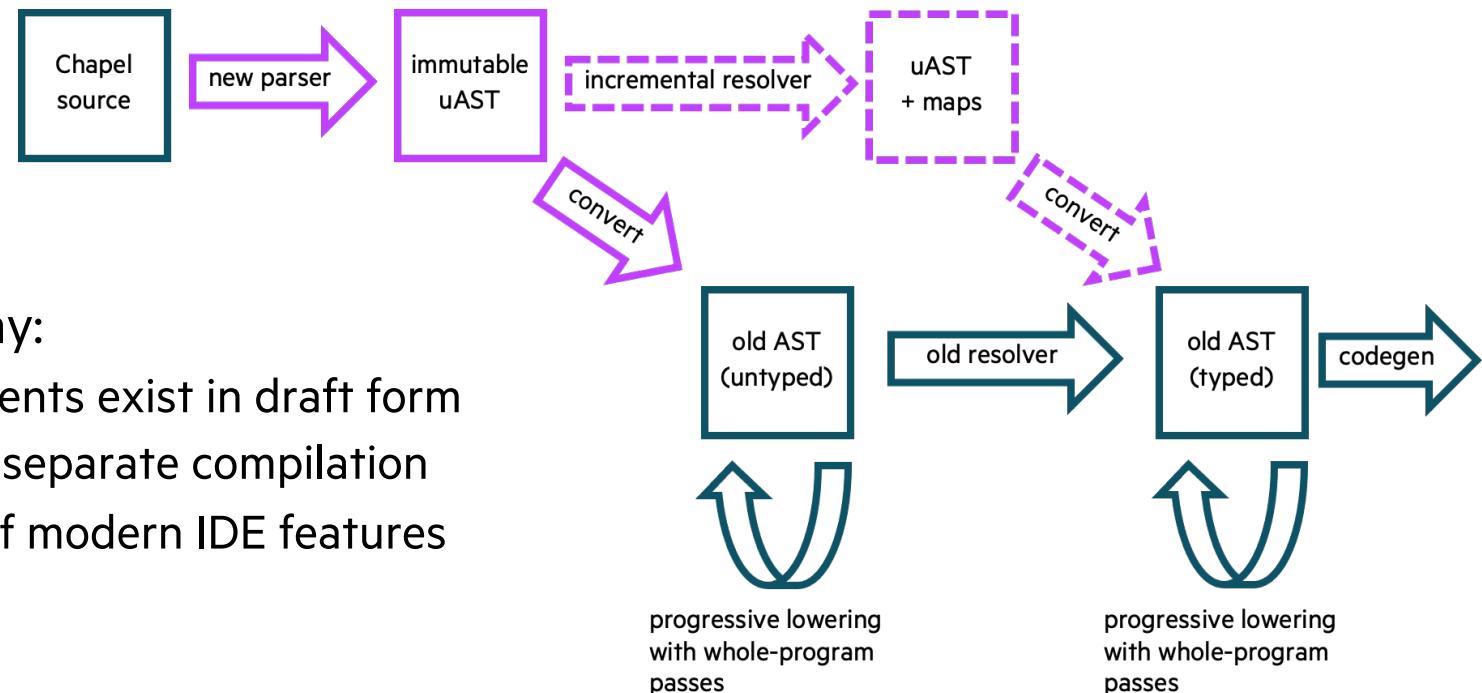
## This Effort:

- Last year, we kicked off an effort to massively rearchitect it, to address these lacks:
  - better user experience in terms of speed and errors
  - easier for developers to start contributing to
  - more flexible and capable:
    - separate compilation / incremental recompilation
    - better support for tools
    - dynamic evaluation of code
    - ...



# DYNO HIGHLIGHTS SINCE CHIUW 2022

- Key elements of ‘dyno’ are now used in the Chapel release:
  - **1.27.0:** its parser and AST became the default
  - **1.28.0:** ‘chpldoc’ was rewritten to use its front-end library
  - **1.29.0:** its framework for improved error messages came online
  - **‘main’:** its scope resolver is now used by default (and will be in this month’s 1.31.0 release)



- Other key elements are well underway:
  - **type/call resolver:** the major components exist in draft form
  - **AST save/restore:** a key step toward separate compilation
  - **Chapel language server:** in support of modern IDE features



# **COMPILING CHAPEL TO GPUS**



# STREAM TRIAD USING GPUS AND CPUS: CHI UW 2022 (SINGLE-LOCALE)

stream-ep.chpl

```
config const n = 1_000_000,  
       alpha = 0.01;
```

```
cobegin {  
    coforall gpuid in 1..numGPUs do on here.getChild(gpuid) {  
        var A, B, C: [1..n] real;  
        A = B + alpha * C;  
    }  
    {  
        var A, B, C: [1..n] real;  
        A = B + alpha * C;  
    }  
}
```

'cobegin { ... }' creates a task per child statement

one task creates GPU tasks

the other runs the multi-CPU triad

This program uses all CPUs and GPUs  
on a single compute node

# STREAM TRIAD USING GPUS AND CPUS: TODAY (SINGLE-LOCALE)

stream-ep.chpl

```
config const n = 1_000_000,  
        alpha = 0.01;  
  
cobegin {  
    coforall gpu in here.gpus do on gpu {  
        var A, B, C: [1..n] real;  
        A = B + alpha * C;  
    }  
    {  
        var A, B, C: [1..n] real;  
        A = B + alpha * C;  
    }  
}
```

Improved syntax for GPU sublocales

This program uses all CPUs and GPUs  
on a single compute node

# STREAM TRIAD USING GPUS AND CPUS: TODAY (MULTI-LOCALE)

stream-ep.chpl

```
config const n = 1_000_000,  
        alpha = 0.01;  
  
coforall loc in Locales do on loc {  
    cobegin {  
        coforall gpu in here.gpus do on gpu {  
            var A, B, C: [1..n] real;  
            A = B + alpha * C;  
        }  
        {  
            var A, B, C: [1..n] real;  
            A = B + alpha * C;  
        }  
    }  
}
```

Support for multi-locale GPU programs

This program uses all CPUs and GPUs  
across *all* of our compute nodes

# CHAPEL ON GPUS: PROGRESS SINCE CHIUW 2022

**Status:** Lots of progress in the past year:

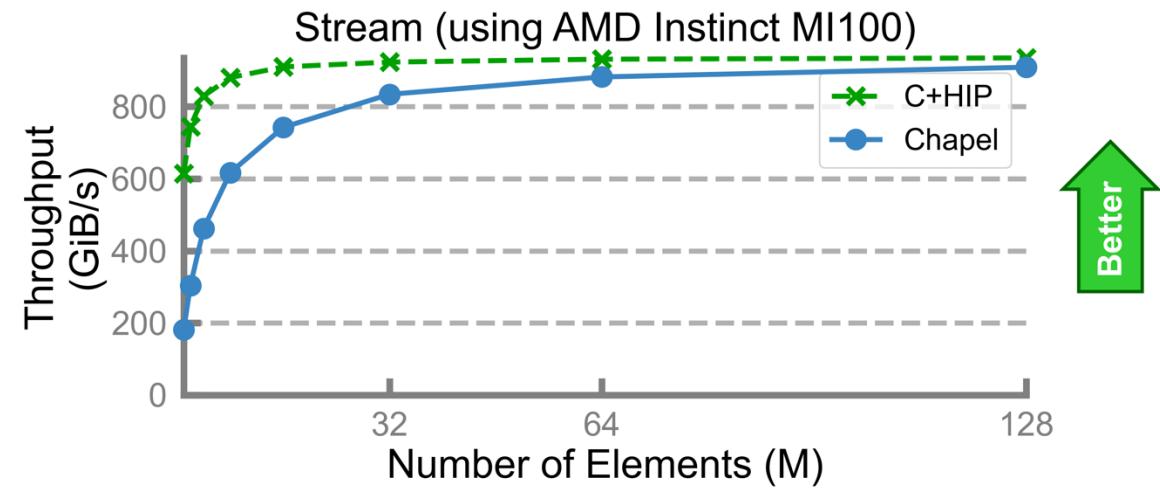
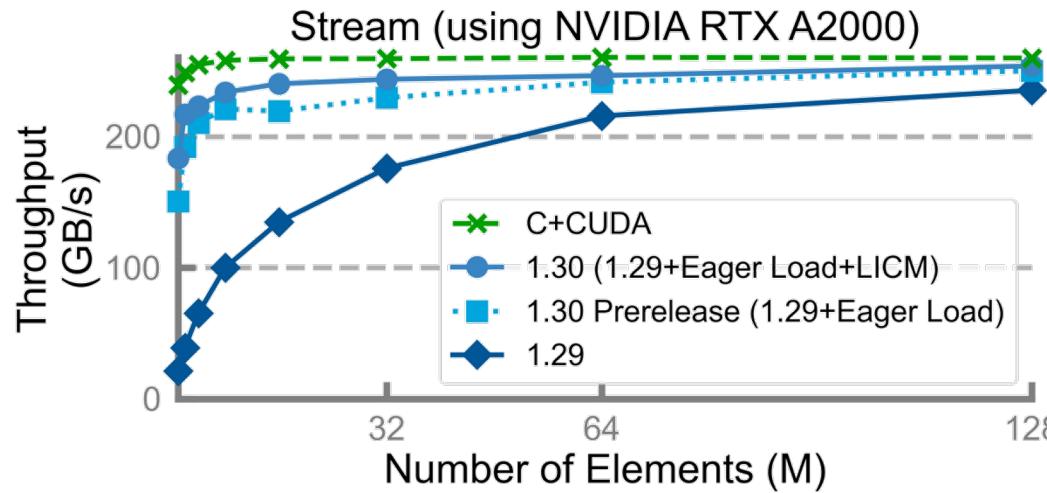
- **Mar 2022** (v1.26): first multi-GPU NVIDIA runs
- **CHIUW 2022**
- **Jun 2022** (v1.27): first multi-locale, multi-GPU NVIDIA runs
- **Dec 2023** (v1.29): first heroic AMD runs
- **Mar 2023** (v1.30): first multi-GPU AMD runs
- **Jun 2023** (v1.31): first multi-locale, multi-GPU AMD runs



Throughout this time, ongoing improvements  
in generality, features, and performance



# STREAM TRIAD: GPU PERFORMANCE VS. REFERENCE VERSIONS



Better

Performance vs. reference versions has become increasingly competitive over the past 6 months

More about GPU Computing with Chapel in Session 5 today (1:35 PDT)



## **LANGUAGE AND LIBRARIES**



# CHAPEL 2.0

---

## Background:

- For the past several years, we have been working toward a forthcoming Chapel 2.0 release
- Intent: stop making backward-breaking changes to core language and library features

## Status as of CHI UW 2022:

- Major language-related changes were considered to have largely wound down (ha!)
- Primary remaining effort was on stabilizing the standard libraries



# CHAPEL 2.0 LIBRARY STABILIZATION

Status: Visualized

	Builtins	ChplConfig	List	Map	Set	FileSystem	IO	Path	Reflection	Types	BigInteger	Math/AutoMath	Random	Barriers	CTypes	Subprocess	Sys	SysBasic	SysError	DateTime	Regex	Time	Version	String / Bytes	Ranges	Domains	Arrays	Shared / Owned	Errors	Memory.MoveInitialization	Locales	SyncVar	Atomics
1.26	✓																																
1.27	✓																																
1.28	✓																																



**Stable**



**Progress**



**Review Started**



# CHAPEL 2.0 LIBRARY STABILIZATION

Status: Visualized

	Builtins	ChplConfig	List	Map	Set	FileSystem	IO	Path	Reflection	Types	BigInteger	Math/AutoMath	Random	Collectives	CTypes	Subprocess	Sys	SysBasic	SysError	DateTime	Regex	Time	Version	String / Bytes	Ranges	Domains	Arrays	Shared / Owned	Errors	MemMove	Locales	SyncVar	Atomics
1.28	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
1.29	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗			
1.30	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗			



**Stable**



**Progress**



**Review Started**



## NEW LANGUAGE AND LIBRARY FEATURES

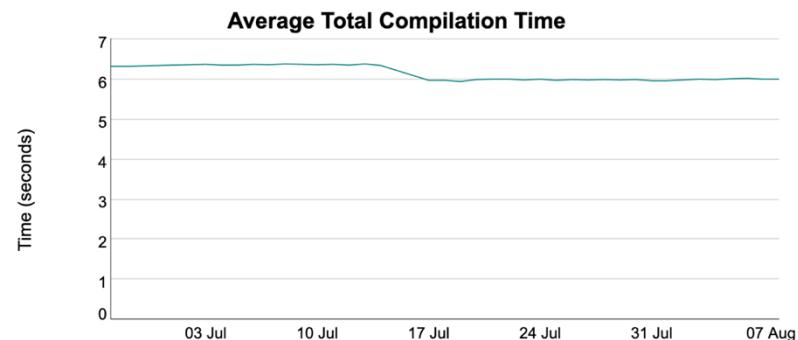
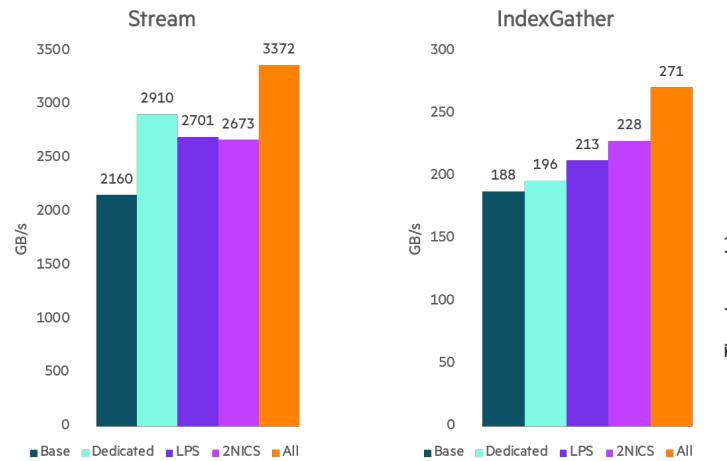
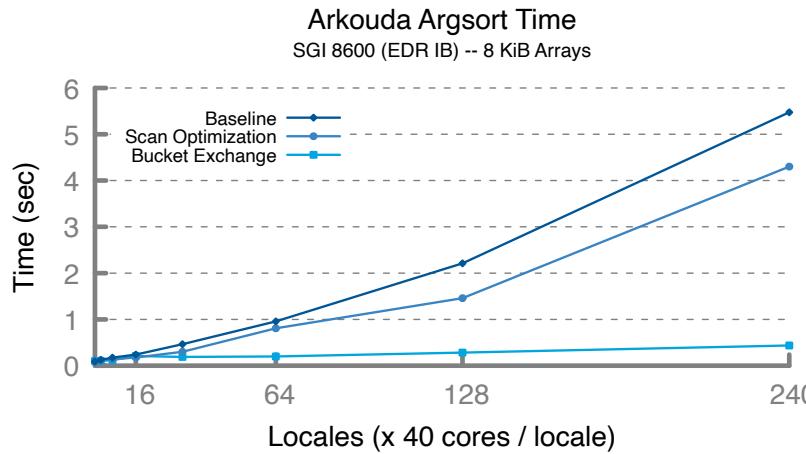
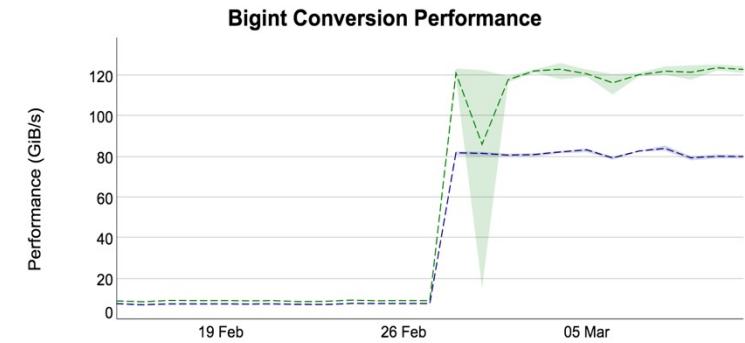
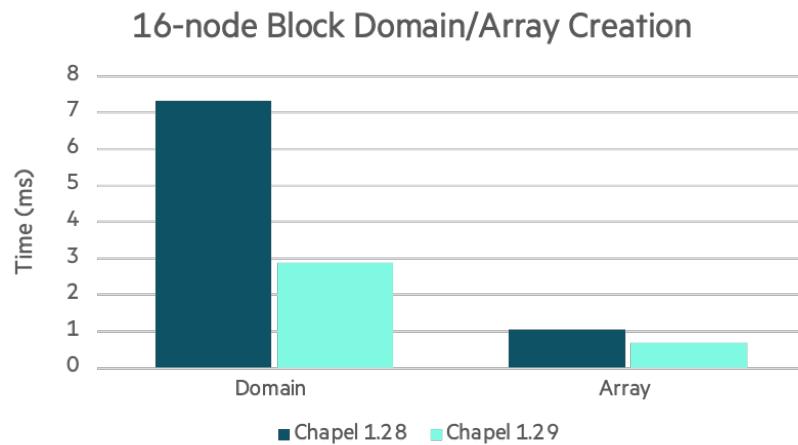
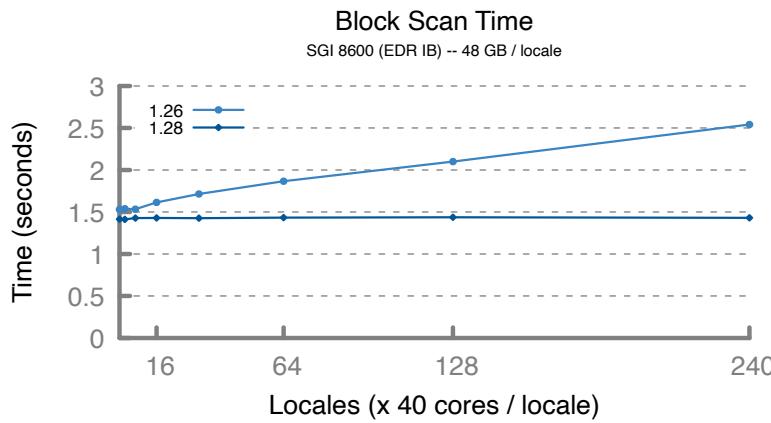
---

- **@attributes:** for embedding information in code outside the language
- **'Communication' module:** for single-sided puts/gets across locales
- **new first-class function syntax:** more aligned with Chapel's procedures
- **weak class pointers:** for use with 'shared'-based classes
- **throwing initializers:** for initializers whose post-field-init bodies may hit errors
- ...

# **PERFORMANCE OPTIMIZATIONS**



# PERFORMANCE OPTIMIZATIONS



# **OUTREACH**



# CHAPEL BLOG

In December, we did a soft-launch of the Chapel Language Blog: <https://chapel-lang.org/blog/>



About Chapel Website Featured Series Tags Authors All Posts

## Featured Articles

### ★ Welcome to the Chapel blog!

Posted on November 30, 2022

An introduction to the Chapel blog, and our intentions and plans for it.

### ★ Advent of Code 2022: Twelve Days of Chapel

Posted on November 30, 2022

The Chapel team's plan for blogging during Advent of Code 2022.

### ★ Announcing Chapel 1.30.0!

Posted on March 23, 2023

A summary of highlights from the March 2023 release of Chapel 1.30.0

### ★ NetCDF in Chapel, Part 1: Interfacing with the C Library

Posted on April 26, 2023

An introduction to C interoperability in Chapel using the NetCDF library



About Chapel Website Featured Series Tags Authors All Posts

### Announcing Chapel 1.30.0!

Posted on March 23, 2023.

Tags: Release Announcements

By Brad Chamberlain

The Chapel developer community is pleased to announce the release of Chapel version 1.30.0! To obtain a copy, please refer to the [Downloading Chapel](#) page on the Chapel website.

#### Highlights of Chapel 1.30.0

##### @Attributes

Chapel 1.30.0 makes good on a longstanding intention to add a generalized attribute capability to the language. These attributes are designed to convey information to the compiler—or other tools—in a way that is integrated with the source code, extensible, and independent of keyword-based language features.

At present, a small set of attributes is supported. In particular, there are a few attributes that can be used to characterize the stability of a feature, as well as a `chpldoc` attribute for suppressing the documentation for a particular declaration. As an example, the attributes on the following procedure will generate a deprecation warning for any calls to `foo()` while also ensuring that `chpldoc` does not generate documentation for `foo()`:

```
1  @deprecated("foo() is deprecated; please use 'newFoo()' instead")
2  @chpldoc.nodoc
3  proc foo() {
4      writeln("In foo()");
5  }
```

Future versions of Chapel will expand upon this initial set of attributes. To learn more about the current support, refer to the [Attributes in Chapel](#) technical note.



## Chapel Language Blog

About Chapel Website Featured Series Tags Authors

## Authors

The following authors have written articles on this blog:



Brad Chamberlain



Daniel Fedorin



Jeremiah Corrado



Michelle Strout



Scott Bachman



# The 6th Annual Parallel Applications Workshop, Alternatives To MPI+X

Monday, November 13, 2023

Held in conjunction with SC23



**Deadline:** July 24, 2023

**Submission Styles:** Papers / Talks

# **WRAPPING UP**



# THE CHAPEL TEAM AT HPE, JUNE 2023



# WHAT'S NEXT?

---

- **Continue with Quarterly Releases:** June, September, December, March
  - September's Chapel 1.32 will be a release candidate for Chapel 2.0
- **HPE Cray EX:** More Benchmarking and Tuning
- **Dyno:** Have it take over resolution of Calls and Types
- **GPUs:** More Features and Performance
- **Blog:** Hard-Launch
- **User Support and Outreach**
- **Performance and Feature Improvements**
- ...

## IN MEMORIAM

- Mike Merrill passed away on November 8<sup>th</sup>
- Mike was the chief architect and developer of Arkouda, as well as a friend to many on the Chapel project



### Arkouda: NumPy-like arrays at massive scale backed by Chapel

Michael Merrill\*, William Reus†, and Timothy Neumann‡

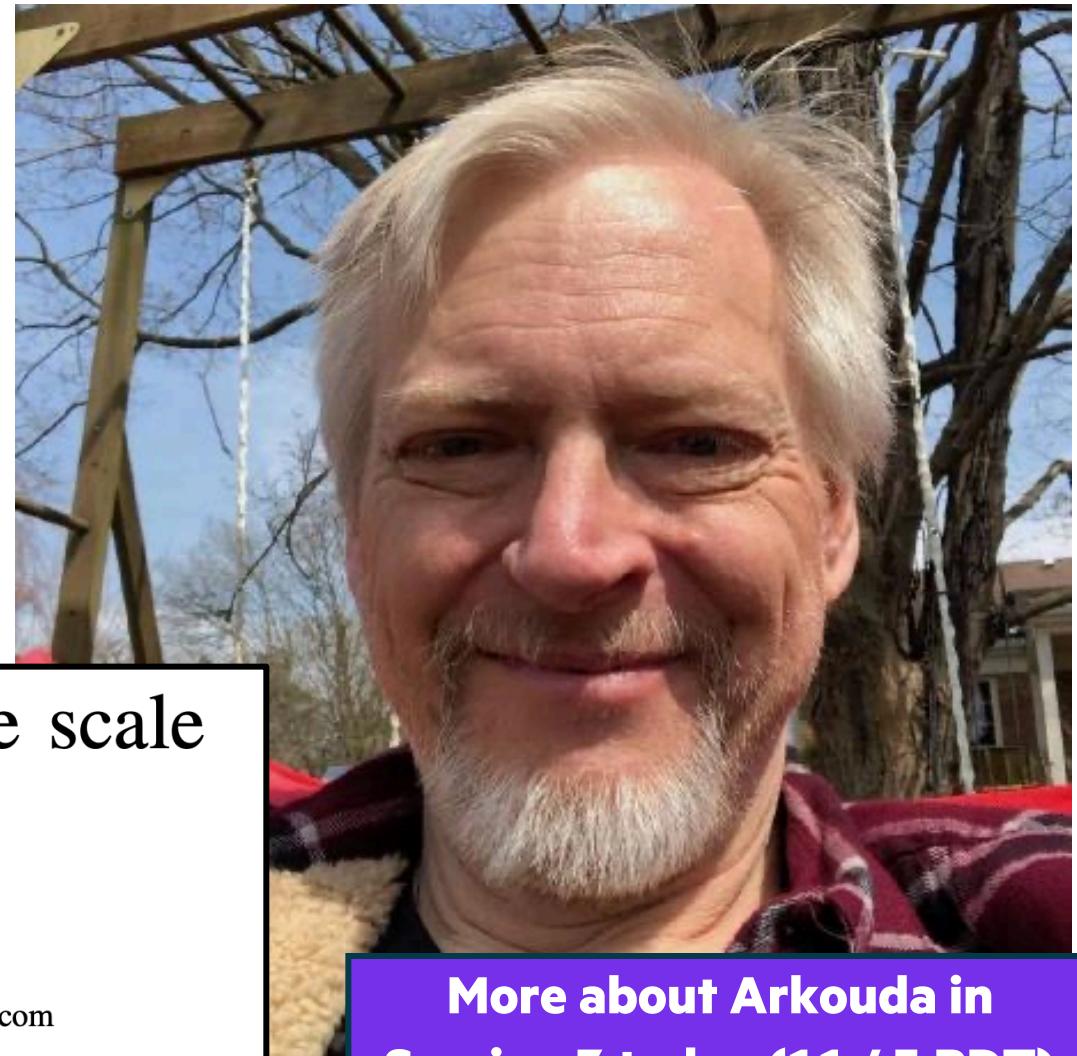
U.S. Department of Defense Washington DC, USA

Email: \*mhmerill@mac.com, †reus@post.harvard.edu, ‡timothyneumann1@gmail.com

Mike's obituary is online at: <https://www.donaldsonclarksville.com/obituary/Michael-Merrill>

## IN MEMORIAM

- Mike Merrill passed away on November 8<sup>th</sup>
- Mike was the chief architect and developer of Arkouda, as well as a friend to many on the Chapel project



### Arkouda: NumPy-like arrays at massive scale backed by Chapel

Michael Merrill\*, William Reus†, and Timothy Neumann‡

U.S. Department of Defense Washington DC, USA

Email: \*mhmerill@mac.com, †reus@post.harvard.edu, ‡timothyneumann1@gmail.com

**More about Arkouda in  
Session 3 today (11:45 PDT)**

# CHAPEL RESOURCES

**Chapel homepage:** <https://chapel-lang.org>

- (points to all other resources)

## Social Media:

- Blog: <https://chapel-lang.org/blog/>
- Twitter: [@ChapelLanguage](https://twitter.com/ChapelLanguage)
- Facebook: [@ChapelLanguage](https://facebook.com/ChapelLanguage)
- YouTube: <https://www.youtube.com/c/ChapelParallelProgrammingLanguage>

## Community Discussion / Support:

- Discourse: <https://chapel.discourse.group/>
- Gitter: <https://gitter.im/chapel-lang/chapel>
- Stack Overflow: <https://stackoverflow.com/questions/tagged/chapel>
- GitHub Issues: <https://github.com/chapel-lang/chapel/issues>



The Chapel Parallel Programming Language

**What is Chapel?**

Chapel is a programming language designed for productive parallel computing at scale.

**Why Chapel?** Because it simplifies parallel programming through elegant support for:

- distributed arrays that can leverage thousands of nodes' memories and cores
- a global namespace supporting direct access to local or remote variables
- data parallelism to trivially use the cores of a laptop, cluster, or supercomputer
- task parallelism to create concurrency within a node or across the system

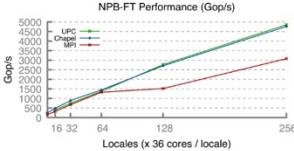
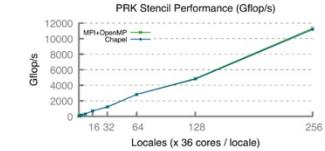
**Chapel Characteristics**

- **productive:** code tends to be similarly readable/writable as Python
- **scalable:** runs on laptops, clusters, the cloud, and HPC systems
- **fast:** performance **competes with or beats** C/C++ & MPI & OpenMP
- **portable:** compiles and runs in virtually any \*nix environment
- **open-source:** hosted on [GitHub](#), permissively licensed

**New to Chapel?**

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

- watch an [overview talk](#) or browse its [slides](#)
- read a [chapter-length introduction](#) to Chapel
- learn about [projects powered by Chapel](#)
- check out [performance highlights](#) like these:



- browse [sample programs](#) or learn how to write distributed programs like this one:

```
use CyclicDist;           // use the Cyclic distribution Library
config const n = 100;      // use -n=<val> when executing to override this default
forall i in Cyclic.createDomain(1..n) do
    writeln("Hello from iteration ", i, " of ", n, " running on node ", here.id);
```

# **THANK YOU**

---

<https://chapel-lang.org>  
@ChapelLanguage

