



Hewlett Packard
Enterprise

CHIUW 2023: STATE OF THE CHAPEL PROJECT

Brad Chamberlain

June 2, 2023

WELCOME TO THE 10TH ANNUAL CHIUW WORKSHOP!



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

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

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 LITS
Claudia Fohry (Universität Kassel, Germany), Jens Breitbart (Technische Universität München, Germany)
9:30 - 10:00: Evaluating Next Generation PGAS Languages for Computational Chemistry
Daniel Chavarria-Miranda , Sriram Krishnamoorthy , Joseph Manzano , Abhinav Vishnu (Pacific Northwest National Laboratory)
10:00 - 10:30: Break
Technical Talks: Language Extensions
10:30 - 11:00: Programmer-Guided Reliability in Chapel
David E. Borzhilov , Wael R. Elwasif , Christos Kartsaklis , Seoyang Lee , Tiffany M. Mintz (Oak Ridge National Laboratory)
11:00 - 11:30: Towards Interfaces for Chapel
Chris Watkins , Jeremy G. Siek (Indiana University)
Technical Talks: Compiler Optimizations
11:30 - 12:00: Affine Loop Optimizations using Modular Linearization in Chapel
Arson Sharma , Bajdev Barak (University of Maryland), Michael Ferguson (Laboratory for Telecommunication Sciences)
12:00 - 1:00: Lunch (on your own)
Invited Talk: Robert Harrison
1:00 - 1:45: Walking to the Chapel
Robert Harrison (Stony Brook University/Brookhaven National Laboratory)
Abstract: 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 HPC's 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 seeding collaborations.
Technical Talks: Compiler Optimizations (continued)
1:45 - 2:15: LLVM Optimizations for PGAS Programs
Abhiru Hagathi , Rishi Surendran , Jinghui Zhao , Vivek Sarkar (Rice University), Michael Ferguson (Laboratory for Telecommunication Sciences)
Technical Talks: Runtime Improvements
2:15 - 2:45: Opportunities for Integrating Tasking and Communication Layers
Dylan T. Stark , Brian W. Barrett (Sandia National Laboratories)
2:45 - 3:15: Caching in an Aggregation
Michael Ferguson (Laboratory for Telecommunication Sciences)
3:15 - 3:30: Break
Community/Panel Discussion
3:30 - 4:30: Anyone who is interested

CHI UW: Chapel Implementers and Users Workshop

<http://chapel.cray.com/CHI UW.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 CHI UW steering committee, we hope to see you at CHI UW!

WELCOME TO THE 10TH ANNUAL CHIUW WORKSHOP!



CHIUIW 2014

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

Attend CHIUIW on Friday and hear
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 LUTS

Claudia Fohry (Universität Kassel, Germany), Jens Breitbart (Technische Universität M

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

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

10:00 - 10:30: Break

Technical Talks: Language Extensions

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

David E. Borzhilov, Wael R. Elwasif, Christos Kartarakis, Seyong Lee, Tiffany M. Mintz

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

Chris Walke, Jeremy G. Siek (Indiana University)

Technical Talks: Compiler Optimizations

11:30 - 12:00: Affine Loop Optimizations using Modular Linearization in Chapel

Ajoun Sharmar, Rajeev Barua (University of Maryland), Michael Ferguson (Laboratory

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

Invited Talk: Robert Harrison

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

Robert Harrison (Stony Brook University/Brookhaven National Laboratory)

Abstract: MADNESS is a general-purpose numerical environment that sits upon a scal
projects including Charm++ and the HPC's programming languages, including Chapel;
said to Chapel's architects several times, maintaining our own runtime is just as unlik
that home Chapel? I'll give you some flavor of what MADNESS does and how, with th

Technical Talks: Compiler Optimizations (continued)

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

Abhinav Hagathi, Rishi Surendran, Jisheng Zhao, Vivek Sarkar (Rice University), Michae

Technical Talks: Runtime Improvements

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

Dylan T. Stark, Brian W. Barrett (Sandia National Laboratories)

2:45 - 3:15: Caching in an Aggregation

Michael Ferguson (Laboratory for Telecommunication Sciences)

3:15 - 3:30: Break

Community/Panel Discussion

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



Workshop

with [IPDPS 2014](#), will
ps designed to bring
her to report on work
Attendance is open to
er to someone simply

On this day

9 years ago



Brad Chamberlain is with Lydia Duncan.

May 23, 2014 · 2

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/CHIUIW.html>

CHAPEL'S TURNING 20?!?

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

Cray Response on Software Productivity
Brad Chamberlain
Thomas Sterling
Terry Greyzck
The Cray Cascade Project

Overheard at SC2002

- ◆ “Will any of the HPCS participants be working on new parallel languages?”
 - Users still don't have the perfect language
 - Or, existing ones haven't come of age yet
- ◆ “Why don't we have something like Matlab yet?”
 - Certain Matlab characteristics resonate with users...

Our Challenge

“How can we support parallel programming such that it achieves Matlab-level approval ratings in the parallel programming arena?”

(And: “Where do we currently fall short?”)

Language Wishlist

- ◆ Graceful ramp for computation:
 - global view with performance model
 - ability to drop to local view for fine-tuning
- ◆ Portable
 - minimize artifacts that reveal architectural factors
 - ◆ message passing, shared memory, vectors, etc.
- ◆ Performance
- ◆ Interoperable with other languages
- ◆ Allows com...
- ◆ Open source
 - workstati...

What else?

- ◆ Need parallel language abstractions for:
 - arrays
 - sparse arrays
 - collections: sets, sequences, hash tables, ...
 - graph-based parallelism
 - abstract, composable operators for each
- ◆ Decoupled Organization:
 - decouple: computation from
 - data structures from
 - memory layout from

Cray PL Strategy

- ◆ Legacy Codes:
 - Continue to support popular languages:
 - ◆ CAF, UPC, MPI, SHMEM, OpenMP
 - Continue to leverage vectorization/parallelization expertise
- ◆ New Codes:
 - Support novel languages to increase productivity:
 - ◆ **high-level:** supports global view, performance model
 - ◆ **low-level:** supports local view; arch. neutral

◆ Cray is interested in novel languages

- ◆ we should continue to strive for increased abstraction
- ◆ yet we're nervous about the acceptance problem

CRAY

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



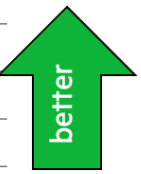
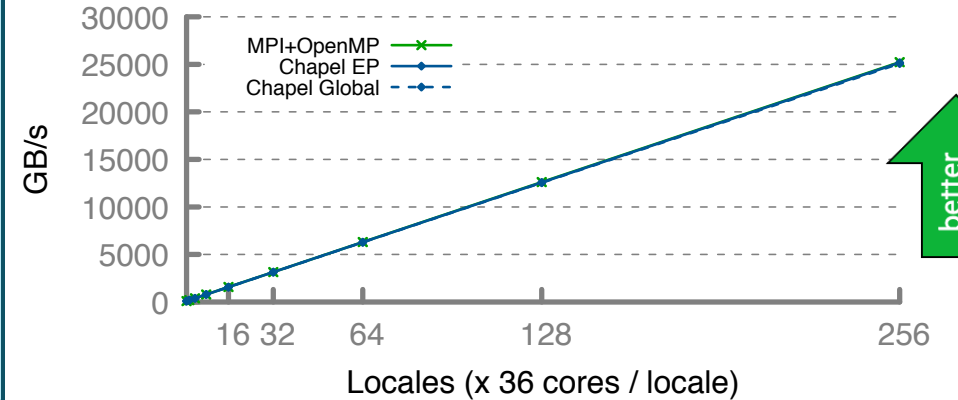
HPC BENCHMARKS: CONVENTIONAL APPROACHES VS. CHAPEL

STREAM TRIAD: C + MPI + OPENMP

```
#include <hpc.h>
#ifdef OPENMP
#include <omp.h>
#endif
static int VectorSize;
static double *a, *b, *c;
int HPC_Stream(HPC_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 = HPC_Stream(params, 0 == myRank);
    MPI_Reduce(&rv, &errCount, 1, MPI_INT, MPI_SUM, 0, comm);
    return errCount;
}
int HPC_Stream(HPC_Params *params, int doTo) {
    register int i;
    double scalar;
    VectorSize = HPC_LocalVectorSize(params, 3, sizeof(double), 0);
    a = HPC_XMALLOC(double, VectorSize);
    b = HPC_XMALLOC(double, VectorSize);
    c = HPC_XMALLOC(double, VectorSize);
    if (doTo) {
        for (i=0; i<VectorSize; i++)
            a[i] = b[i] + c[i];
    }
    return 0;
}
```

```
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;
```

STREAM Performance (GB/s)



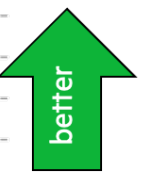
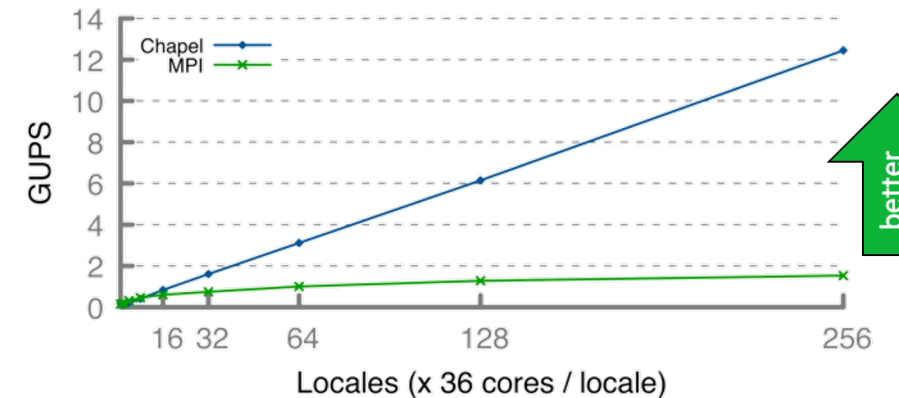
HPC RA: MPI KERNEL

```
/* Perform update to matrix. The order equivalent is
 * A[i] = A[i] + B[i] * C[i]
 * where A, B, and C are matrices of size N.
 * MPI_IncV(LocalBuffer, LocalBuffer, Update, &update,
 * MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &count);
 * MPI_Finalize();
 */
int HPC_RA(HPC_Params *params, int doTo) {
    int myRank, commSize;
    MPI_Comm comm = MPI_COMM_WORLD;
    MPI_Comm_size(comm, &commSize);
    MPI_Comm_rank(comm, &myRank);
    MPI_IncV(LocalBuffer, LocalBuffer, Update, &update,
    MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &count);
    MPI_Finalize();
}
int HPC_RA(HPC_Params *params, int doTo) {
    register int i;
    double scalar;
    VectorSize = HPC_LocalVectorSize(params, 3, sizeof(double), 0);
    a = HPC_XMALLOC(double, VectorSize);
    b = HPC_XMALLOC(double, VectorSize);
    c = HPC_XMALLOC(double, VectorSize);
    if (doTo) {
        for (i=0; i<VectorSize; i++)
            a[i] = b[i] + c[i];
    }
    return 0;
}
```

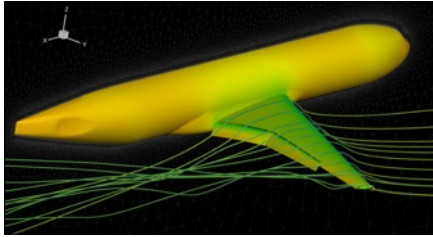
```
... forall (_, r) in zip(Updates, RAStream()) do
    T[r & indexMask].xor(r);
...

```

RA Performance (GUPS)

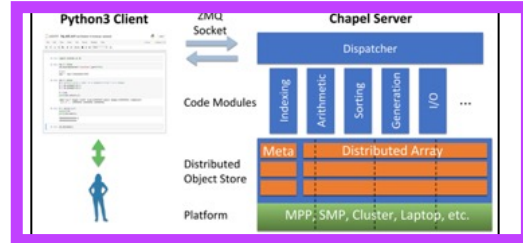


APPLICATIONS OF CHAPEL



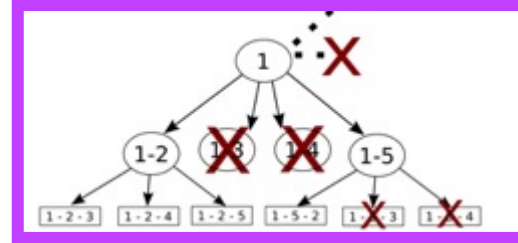
CHAMPS: 3D Unstructured CFD

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



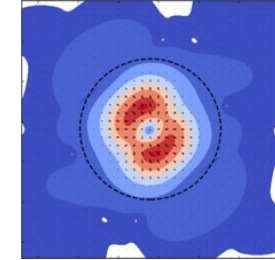
Arkouda: Interactive Data Science at Massive Scale

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



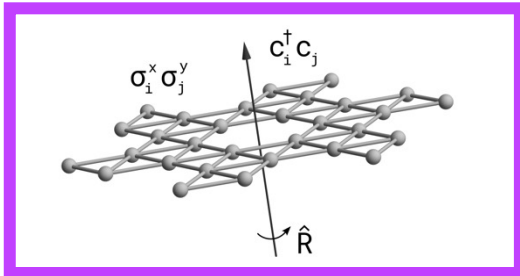
ChOp: Chapel-based Optimization

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



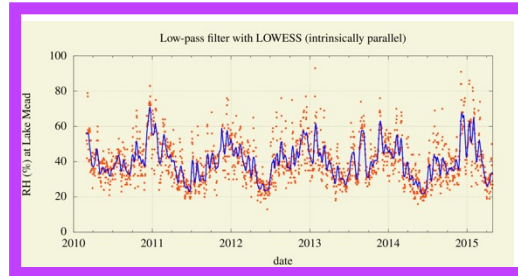
ChpUltra: Simulating Ultralight Dark Matter

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



Lattice-Symmetries: a Quantum Many-Body Toolbox

Tom Westerhout
Radboud University



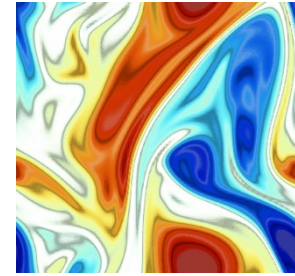
Desk dot chpl: Utilities for Environmental Eng.

Nelson Luis Dias
The Federal University of Paraná, Brazil



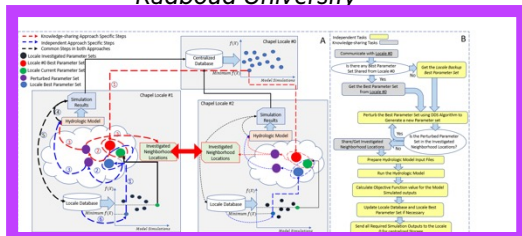
RapidQ: Mapping Coral Biodiversity

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



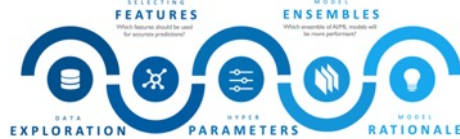
ChapQG: Layered Quasigeostrophic CFD

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

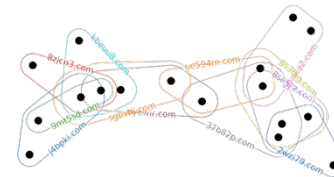


Chapel-based Hydrological Model Calibration

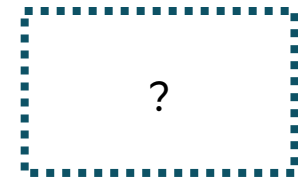
Marjan Asgari et al.
University of Guelph



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**



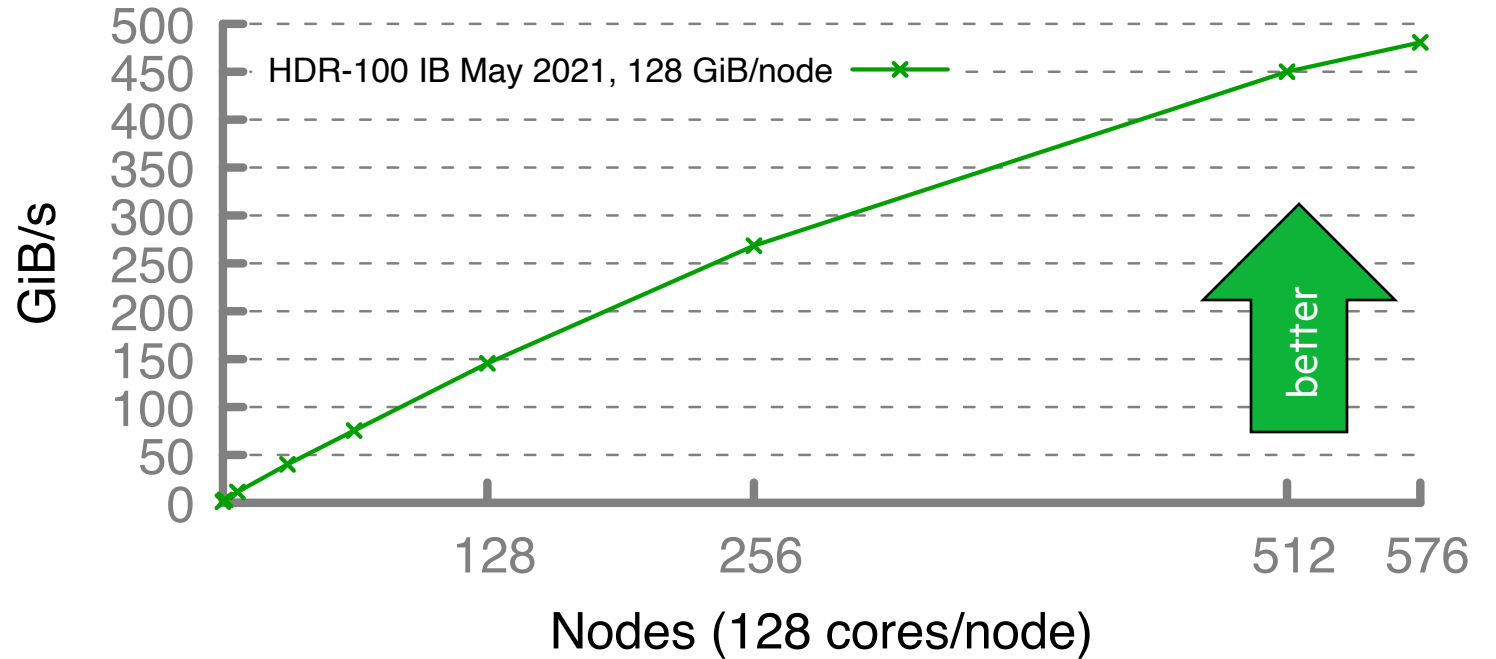
ARKOUDA ARGSORT PERFORMANCE: CHIUW 2022

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)

Arkouda ArgSORT Performance



A notable performance achievement in ~100 lines of Chapel



ARKOUDA ARGSORT PERFORMANCE: TODAY

HPE Apollo (May 2021)



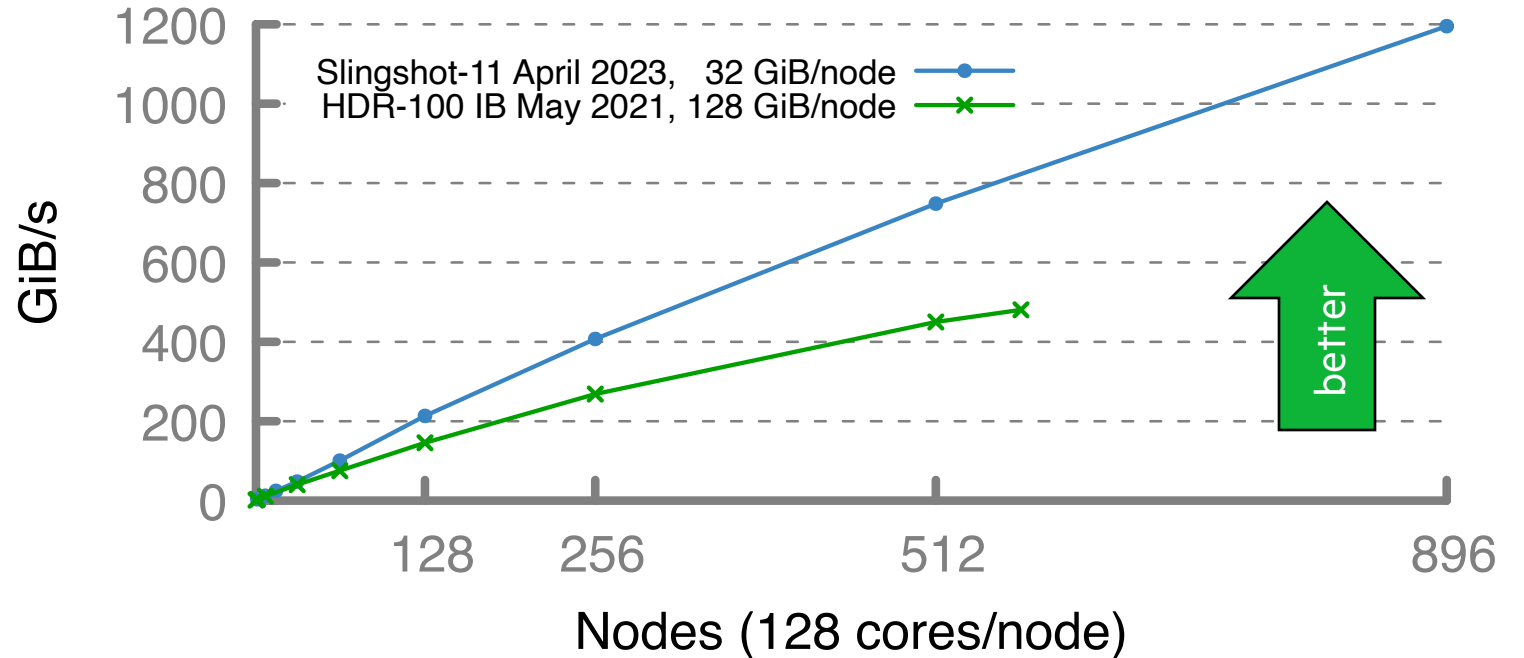
- HDR-100 Infiniband network (100 Gb/s)
- 73,728 cores / 576 nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)

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)

Arkouda ArgSORT Performance



A notable performance achievement in ~100 lines of Chapel



ARKOUDA ARGSORT 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)



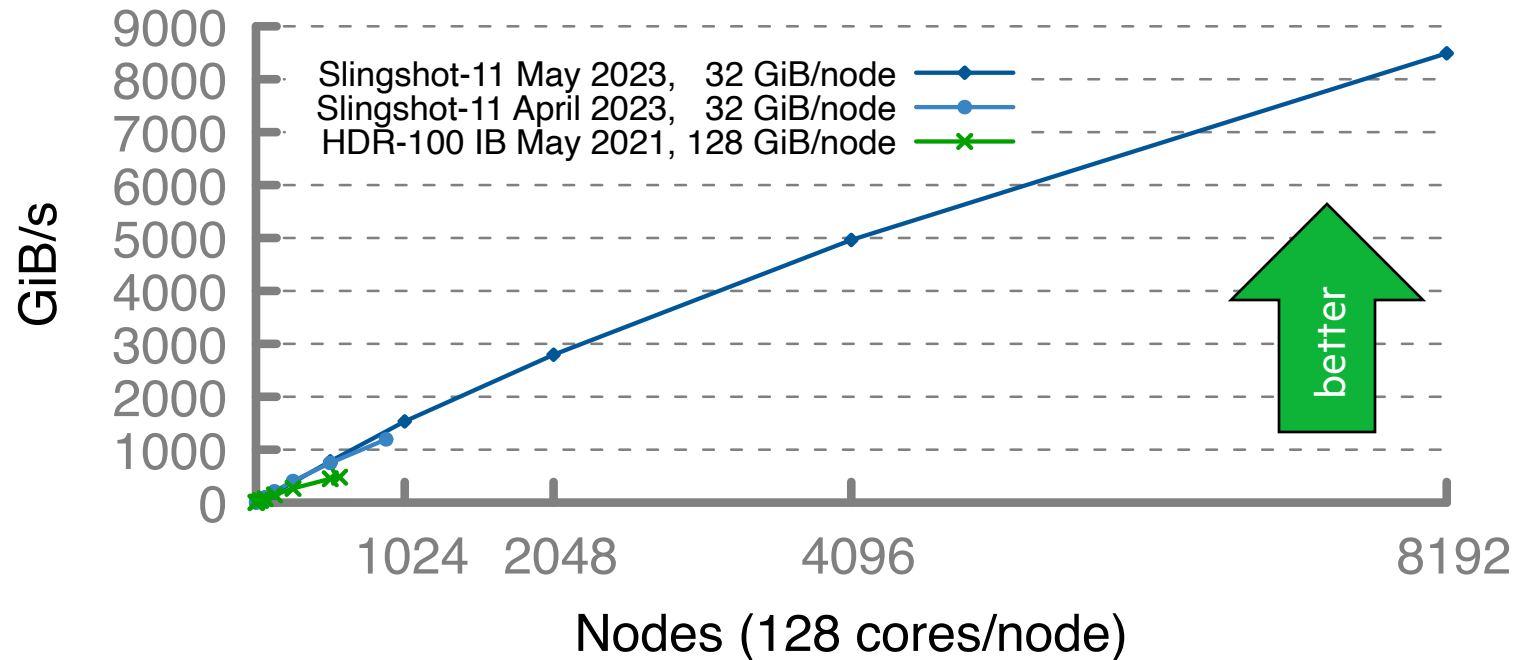
- Slingshot-11 network (200 Gb/s)
- 114,688 cores / 896 nodes
- 28 TiB of 8-byte values
- ~1200 GiB/s (~24 seconds)

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)

Arkouda ArgSORT Performance



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

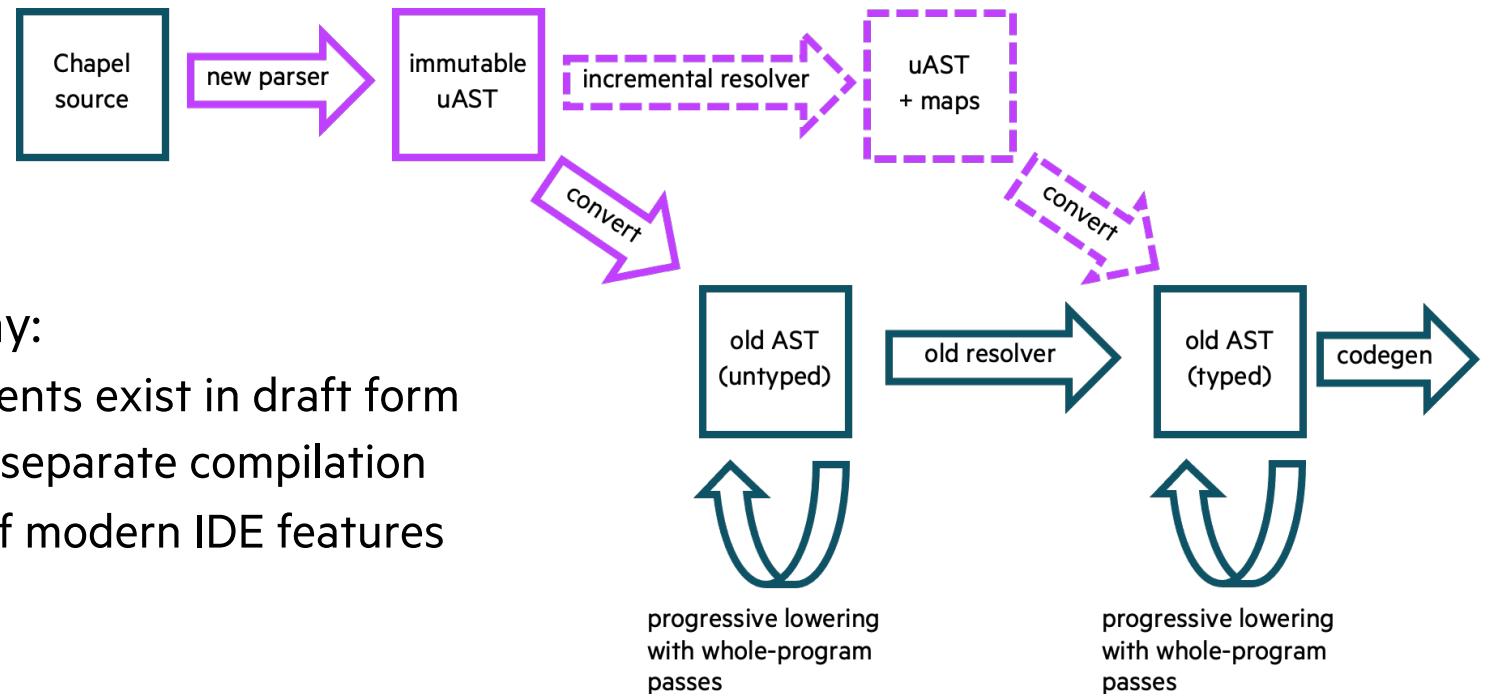
This Effort:

- 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: CHIUW 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 Llocales 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:

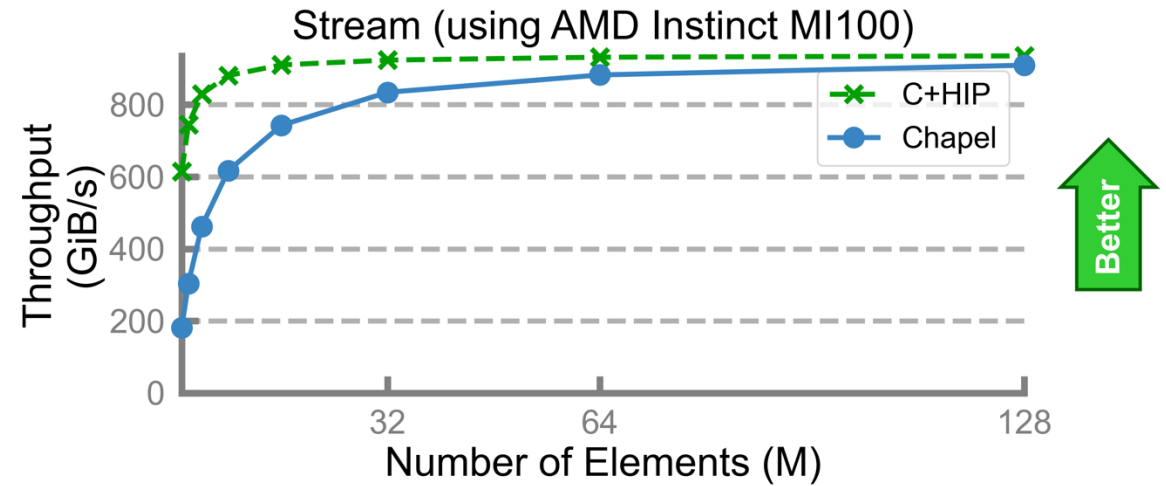
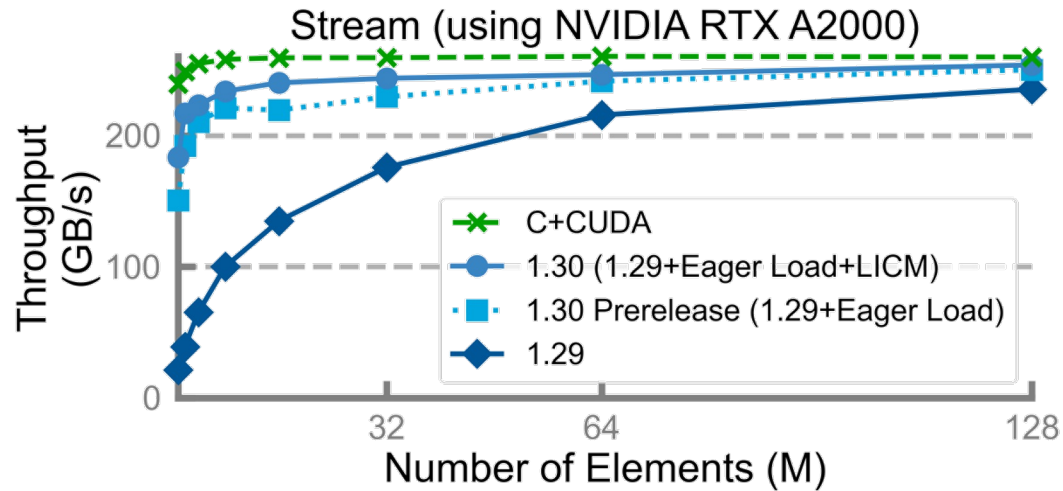
- **Mar 2022** (v1.26): first multi-GPU NVIDIA runs
- **CHIUW 2022**
- **Jun 2022** (v1.27): first multi-locale, multi-GPU NVIDIA runs
- **Dec 2023** (v1.29): first heroic AMD runs
- **Mar 2023** (v1.30): first multi-GPU AMD runs
- **Jun 2023** (v1.31): first multi-locale, multi-GPU AMD runs



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



STREAM TRIAD: GPU PERFORMANCE VS. REFERENCE VERSIONS



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

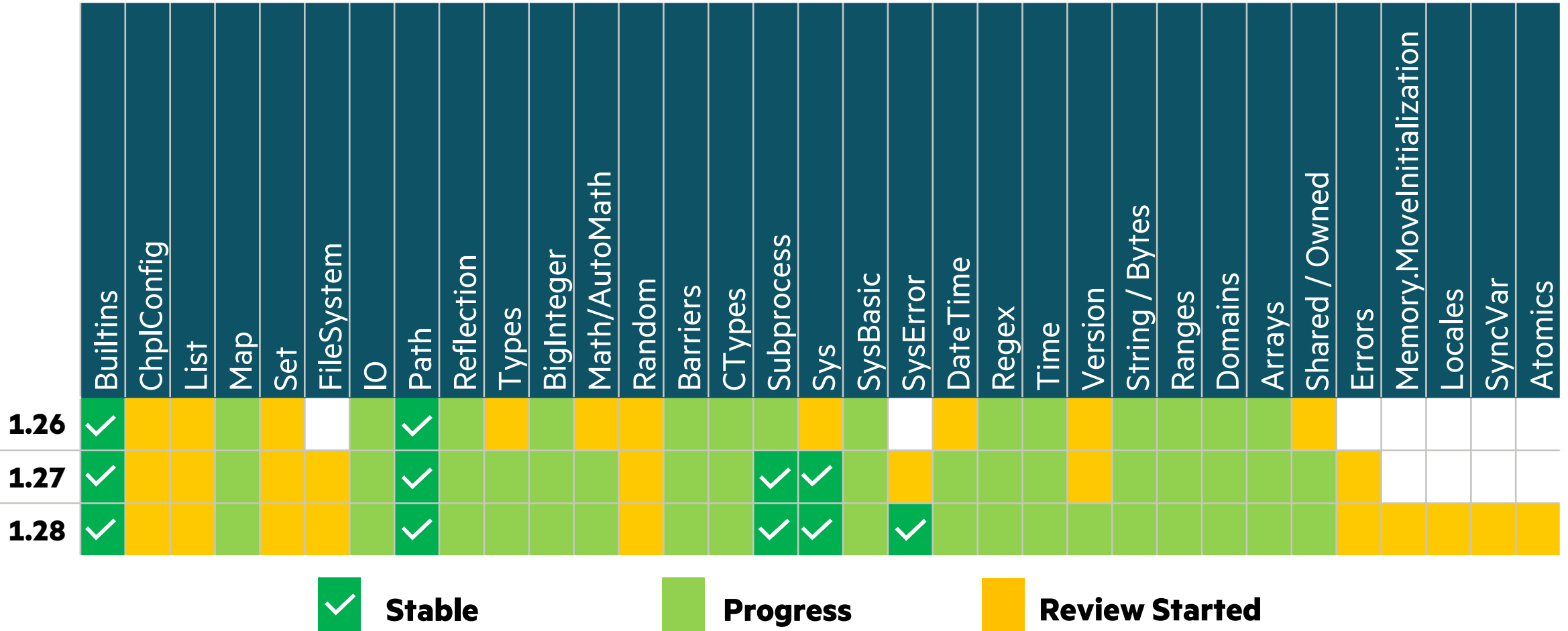
Status as of CHI UW 2022:

- 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



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	✓	Review Started	Review Started	Progress	Review Started	Review Started	Progress	✓	Progress	Progress	Progress	Progress	Review Started	Progress	Progress	✓	✓	Progress	✓	Progress	Progress	Progress	Progress	Progress	Progress	Progress	Progress	Progress	Review Started	Review Started	Review Started	Review Started	Review Started	
1.29	✓	Review Started	Review Started	Progress	Review Started	Progress	Progress	✓	Progress	✓	Progress	Progress	Review Started	Progress	Progress	✓	✓	✓	✓	Progress	Progress	Progress	✓	Progress	Progress	Progress	Progress	Progress	Progress	Progress	Progress	Review Started	Review Started	Review Started
1.30	✓	Review Started	Progress	Progress	Review Started	Progress	Progress	✓	Progress	✓	Progress	Progress	Review Started	Progress	Progress	✓	✓	✓	✓	Progress	✓	Progress	✓	Progress	Progress	Progress	Progress	Progress	Progress	Progress	✓	✓	Review Started	Review Started

✓ **Stable**

Progress **Progress**

Review Started **Review Started**



NEW LANGUAGE AND LIBRARY FEATURES

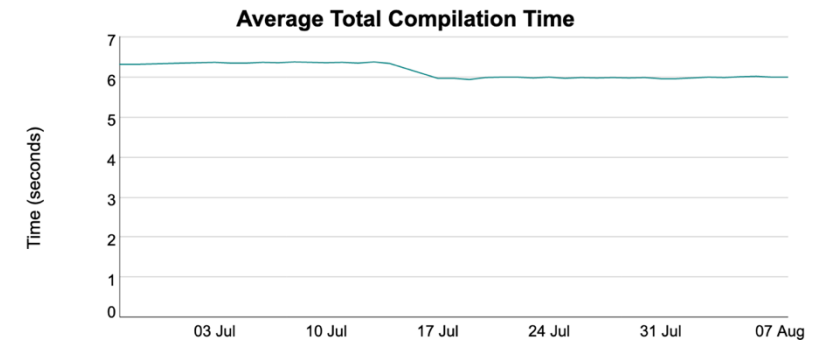
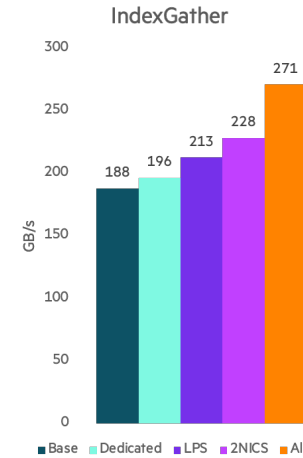
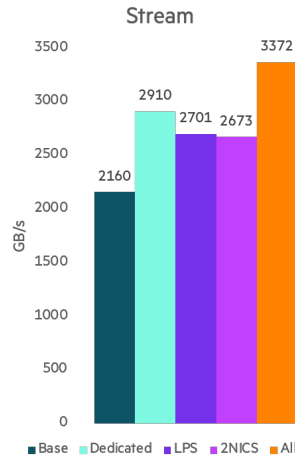
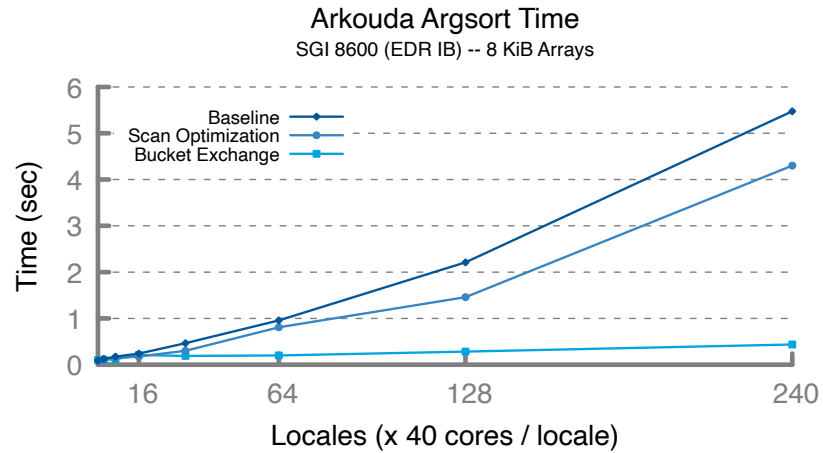
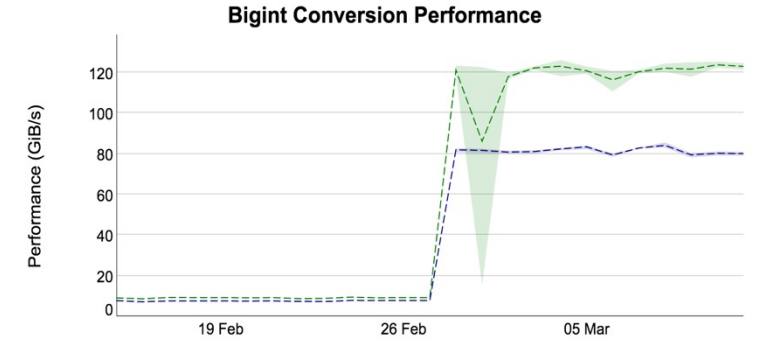
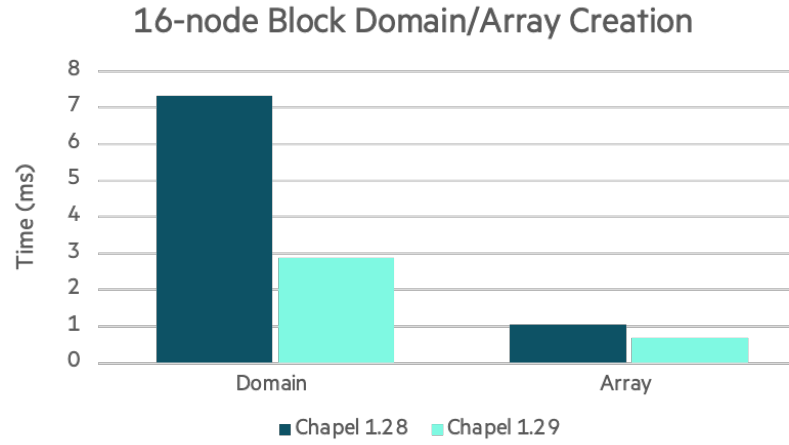
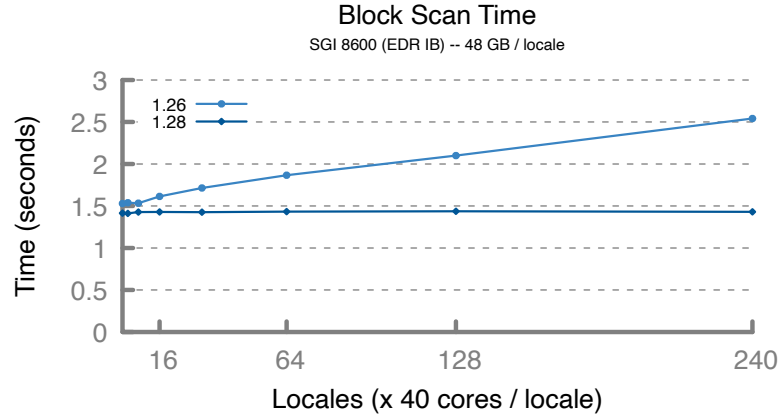
- **@attributes:** for embedding information in code outside the language
- **'Communication' module:** for single-sided puts/gets across locales
- **new first-class function syntax:** more aligned with Chapel's procedures
- **weak class pointers:** for use with 'shared'-based classes
- **throwing initializers:**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



PERFORMANCE OPTIMIZATIONS



PERFORMANCE OPTIMIZATIONS

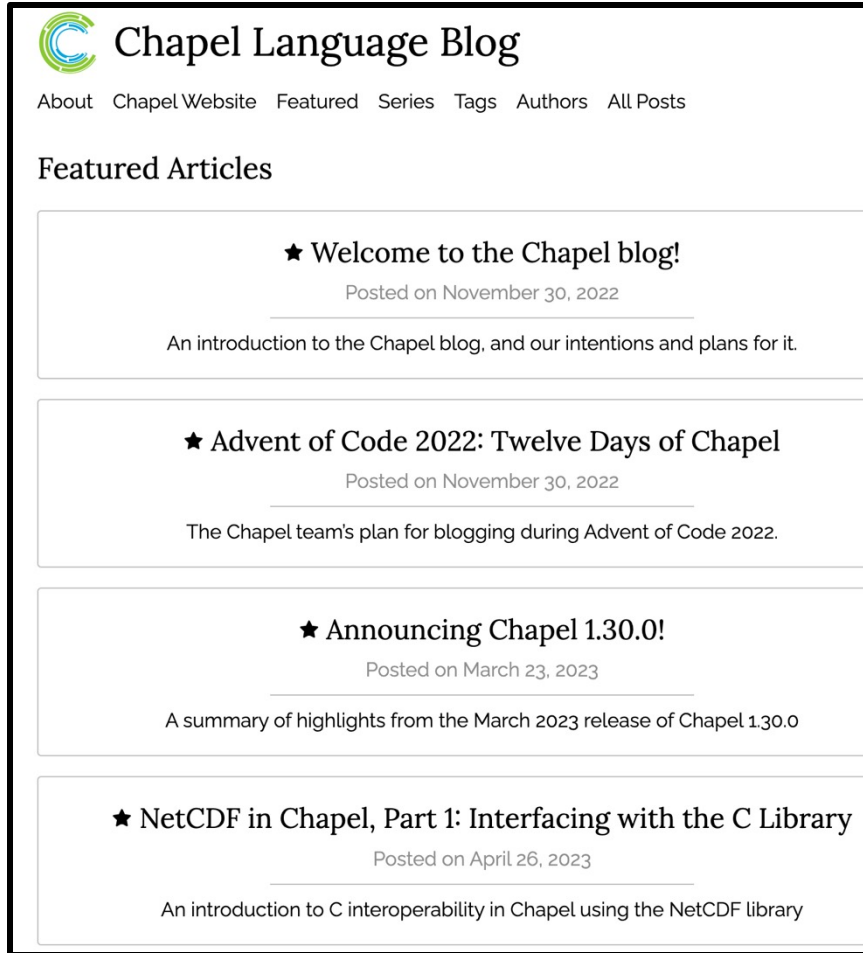


OUTREACH



CHAPEL BLOG

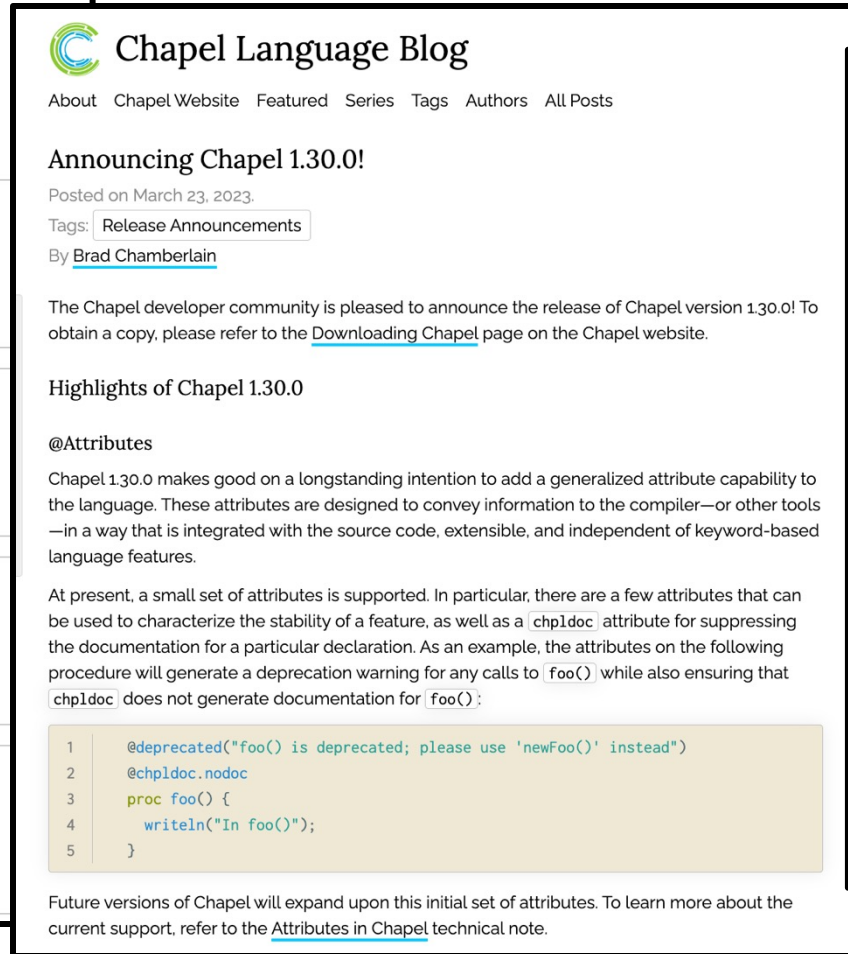
In December, we did a soft-launch of the Chapel Language Blog: <https://chapel-lang.org/blog/>



Chapel Language 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



Chapel Language Blog
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 8th
- Mike was the chief architect and developer of Arkouda, as well as a friend to many on the Chapel project



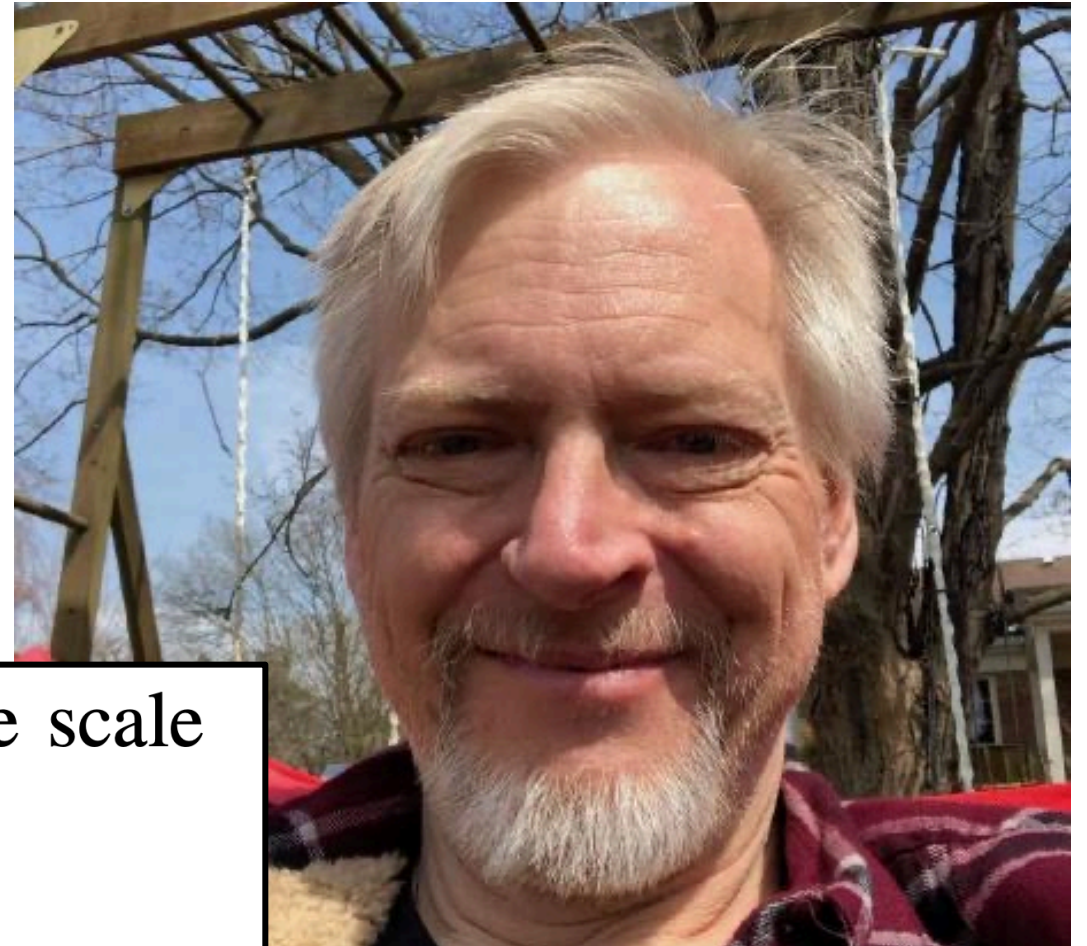
Arkouda: NumPy-like arrays at massive scale backed by Chapel

Michael Merrill*, William Reus[†], and Timothy Neumann[‡]
U.S. Department of Defense Washington DC, USA

Email: *mhmerrill@mac.com, [†]reus@post.harvard.edu, [‡]timothyneumann1@gmail.com

IN MEMORIAM

- Mike Merrill passed away on November 8th
- 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: *mhmerrill@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://www.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

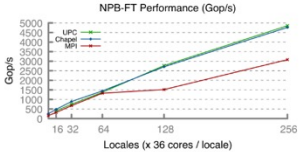
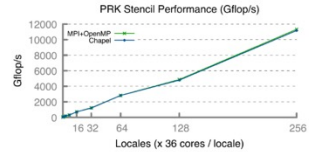
Chapel Characteristics

- **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:



Locales (x 36 cores / locale)	MPI-OpenMP	Chapel
16	~1000	~1000
32	~2000	~2000
64	~4000	~4000
128	~8000	~8000
256	~12000	~12000

Locales (x 36 cores / locale)	LPC	Chapel	SPR
16	~1000	~1000	~1000
32	~2000	~2000	~2000
64	~4000	~4000	~4000
128	~8000	~8000	~8000
256	~12000	~12000	~12000

- 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

