ChapelCon 2024: State of the Chapel Project

Brad Chamberlain
June 7, 2024
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
Chapel 2.0
**Chapel 2.0 has been released!**

**What is Chapel 2.0?**

- A milestone release!
- Stabilizes core language and library features
  - these features should not have breaking changes in the future
- **Released:** March 21, 2024
- Chapel 1.32/1.33 served as release candidates

[Learn more about Chapel 2.0](https://chapel-lang.org/blog/posts/announcing-chapel-2.0/)

---

**Table of Contents**
- What Are People Doing With Chapel?
- A Language Built with Scalable Parallel Computing in Mind
- How to Use Chapel at Scale
- A Framework for Computational Fluid Dynamics
- Coral Biodiversity Computation
- Conclusion and Looking Forward

Today, the Chapel team is excited to announce the release of Chapel version 2.0. After years of hard work and continuous improvements, Chapel shines as an enjoyable and productive programming language for distributed and parallel computing. People with diverse application goals are leveraging Chapel to quickly develop fast and scalable software, including physical simulations, massive data and graph analytics, portions of machine learning pipelines, and more. The 2.0 release brings stability guarantees to Chapel's battle-tested features, making it possible to write performant and elegant code for laptops, GPU workstations, and supercomputers with confidence and convenience.

In addition to numerous usability and performance improvements — including many over the previous release candidate — the 2.0 release of Chapel is stable: the core language and library features are designed to be backwards-compatible from here on. As Chapel continues to grow and evolve, additions or changes to the language should not require adjusting any existing code.
## Chapel 2.0 Library Stabilization: Status as of CHIUW 2023

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builtins</td>
<td>Complete</td>
</tr>
<tr>
<td>ChplConfig</td>
<td>Complete</td>
</tr>
<tr>
<td>List</td>
<td>Complete</td>
</tr>
<tr>
<td>Map</td>
<td>Complete</td>
</tr>
<tr>
<td>Set</td>
<td>Complete</td>
</tr>
<tr>
<td>FileSystem</td>
<td>Complete</td>
</tr>
<tr>
<td>IO</td>
<td>Complete</td>
</tr>
<tr>
<td>Path</td>
<td>Complete</td>
</tr>
<tr>
<td>Reflection</td>
<td>Complete</td>
</tr>
<tr>
<td>Types</td>
<td>Complete</td>
</tr>
<tr>
<td>BigInteger</td>
<td>Complete</td>
</tr>
<tr>
<td>Math/AutoMath</td>
<td>Complete</td>
</tr>
<tr>
<td>Random</td>
<td>Complete</td>
</tr>
<tr>
<td>Collectives</td>
<td>Complete</td>
</tr>
<tr>
<td>CTypes</td>
<td>Complete</td>
</tr>
<tr>
<td>Subprocess</td>
<td>Complete</td>
</tr>
<tr>
<td>Sys</td>
<td>Complete</td>
</tr>
<tr>
<td>SysBasic</td>
<td>Complete</td>
</tr>
<tr>
<td>SysError</td>
<td>Complete</td>
</tr>
<tr>
<td>Regex</td>
<td>Complete</td>
</tr>
<tr>
<td>Time</td>
<td>Complete</td>
</tr>
<tr>
<td>Version</td>
<td>Complete</td>
</tr>
<tr>
<td>String / Bytes</td>
<td>Complete</td>
</tr>
<tr>
<td>Ranges</td>
<td>Complete</td>
</tr>
<tr>
<td>Domains</td>
<td>Complete</td>
</tr>
<tr>
<td>Arrays</td>
<td>Complete</td>
</tr>
<tr>
<td>Shared / Owned</td>
<td>Complete</td>
</tr>
<tr>
<td>Errors</td>
<td>Complete</td>
</tr>
<tr>
<td>MemMove</td>
<td>Complete</td>
</tr>
<tr>
<td>Locales</td>
<td>Complete</td>
</tr>
<tr>
<td>Sync/Single</td>
<td>Complete</td>
</tr>
<tr>
<td>Atomics</td>
<td>Complete</td>
</tr>
<tr>
<td>BitOps</td>
<td>Complete</td>
</tr>
</tbody>
</table>

- **Complete**: The feature is ready for stabilization.
- **Progress**: The feature is being reviewed.
- **Review Started**: The feature is in the review process.
# Chapel 2.0 Library Stabilization: Progress since CHIUW 2023

<table>
<thead>
<tr>
<th></th>
<th>1.30</th>
<th>1.31</th>
<th>1.32</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builtins</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>ChplConfig</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>List</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Map</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Set</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>FileSystem</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>IO</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Path</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Reflection</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Types</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>BigInteger</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Math/AutoMath</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Random</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Collectives</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>CTypes</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Subprocess</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Sys</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>SysBasic</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>SysError</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Regex</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Time</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Version</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>String / Bytes</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Ranges</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Domains</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Arrays</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Shared / Owned</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Errors</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>MemMove</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Locales</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Sync / Single</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>Atomics</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
<tr>
<td>BitOps</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
</tr>
</tbody>
</table>

**Legend:**
- **Complete**: Library has been stabilized.
- **Progress**: Library has been reviewed and is in progress.
- **Review Started**: Library is being reviewed.

**CHIUW 2023**
Chapel 2.0 Language Stabilization: Highlights Since CHIUW 2023

Language:

- Visibility of generic types
- Handling of numeric types, including generics
- ‘range’/‘domain’/array/distribution improvements
- ‘string’/‘bytes’/‘c_string’ stabilization
- ‘sync’/‘single’ stabilization
- Intents: defaults for arrays, return/yield intents, ...
- Protection of special method names
- Lifetimes of temporaries
- Classes: casting, lifetime management, ...
- Implicit ‘param’ to ‘const ref’ conversions
- Made ‘serial’ statements unstable
- Marked ‘local’ statements unstable
- ...

Chapel 2.0: What’s Next?

- **Chapel 2.1**: scheduled for June 27

- **Continue stabilizing features**
  - Prioritize those identified by users / developers
    - e.g., ‘Sort’ module, ‘dmapped’ keyword, ...
    - use ‘--warn-unstable’ to identify unstable features

- **Establish means of making future changes**
  - e.g., create a Chapel language advisory board?
Community Focus
Focus on the Chapel Community

• With 2.0 wrapped up, we have shifted focus toward nurturing the Chapel community
  • Supporting existing users
  • Seeking out new use-cases and users
  • Amplifying our message about Chapel

• This has become our new “all-hands” activity

• Since CHIUW 2023, we have hired our first-ever community manager, Sarah Coghlan

• We’ve also kicked off several initiatives with this community focus in mind:
  • Rebranding CHIUW to ChapelCon, expanding its scope and format
  • Renewed focus on resolving user GitHub issues
    – 140 closed since CHIUW 2023
    – 88 closed the year prior to that
  • Plus…
Welcome to the Chapel Blog (Nov 2022)

14-part “12 Days of Chapel” Series (Dec 2022)

Announcing Chapel 1.29.0! (Dec 2022)

Announcing Chapel 1.30.0! (Mar 2023)

2-part “NetCDF in Chapel” Series (Apr–May 2023)
Chapel Blog: Articles since CHIUW 2023

- Generic Linear Multistep Method Evaluator using Chapel (May)
- Doing Science in Python? Wishing for more speed or scalability? (Apr)
- Chapel's High-Level Support for CPU-GPU Data Transfers and Multi-GPU Programming (Apr)
- Navier-Stokes in Chapel — Introduction (Apr)
- Supercharged Chapel Editor Support (Apr)
- Introducing ChapelCon ’24: The Chapel Event of the Year (Apr)
- Chapel 2.0: Scalable and Productive Computing for All (Mar)
- Changes to Chapel 2.0 Since its First Release Candidate (Feb)
- Comparing Standard Library Sorts: The Impact of Parallelism (Jan)
- Introduction to GPU Programming in Chapel (Jan)
- Announcing Chapel 1.33! (Dec)
- SC23 from the Chapel Language Perspective (Dec)
- Announcing Chapel 1.32! (Sept)
- Announcing Chapel 1.31! (June)
Chapel Social Media

- As with the blog, we’re striving to improve the frequency of our social media posts
  - Goal: a tweet every day or two during the work-week
- Recently launched LinkedIn and Mastadon project accounts—follow us there!
Website Redesign

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:
- 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
- a global namespace supporting direct access to local or remote variables
- GPU programming in a vendor-neutral manner using the same features as above
- distributed arrays that can leverage thousands of nodes’ memories and cores

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 conventional HPC programming models
- portable: compiles and runs in virtually any *nix environment
- open-source: hosted on GitHub, permissively licensed
- production-ready: used in real-world applications spanning diverse fields

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:

---

Massive scale data science, from the comfort of your laptop

```
import arkouda as ak
ak.startup('localhost', 5555)
# Generate two large arrays
a = ak.randint(0, 2**32, 2**10)
b = ak.randint(0, 2**32, 2**10)
# add then
c = a + b
# Sort the array and print first 10 elements
ak.sort(c)
print(c[0:10])
```

---
Other Community-Oriented Things That Are Cooking

- Uptick in talks, tutorials, outreach
- Diversifying modes of obtaining Chapel
  - GitHub Codespaces (already available)
  - Ubuntu and Enterprise 9 packages (already available)
  - deb/rpm packages
  - Spack package
  - AWS AMIs (Amazon Machine Images)?
- Applying to HPSF (High Performance Software Foundation)
- “How-to” YouTube videos
- Chapel educators’ forum
- Quarterly newsletter?
- Community office hours?
- [...Your ideas here...]
  - And your help in growing awareness of Chapel is always appreciated!

Luca speaking today @ 12:00 PT
Chapel Community Survey

• We launched our first-ever Chapel Community Survey
  • We would appreciate if everyone attending ChapelCon were to fill it out
  • (Note that there is also a survey about ChapelCon itself—both would be ideal)
GPUs
GPU Highlights Since CHIUW 2023

- **Improved AMD support**
  - multi-locale + multi-GPU runs now supported
  - features are in parity with NVIDIA

- **CPU-as-device mode**
  - develop CPU+GPU computations without access to a GPU

- **Improved integration with Chapel language**
  - ‘reduce’ expressions and intents
  - ‘foreach’ intents
  - greater use of ‘@attribute’ syntax

- **Plus, performance, scaling, and significant increases in community usage...**
GPU Performance Improvements Since CHIUW 2023

```plaintext
var CpuArr: [1..n] int;
on here.gpus[0] {
  var GpuArr: [1..n] int;
  GpuArr = CpuArr;
  CpuArr = GpuArr;
}
```

**Throughput (Gib/s)**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>86.2</td>
<td>86.3</td>
<td>86.3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>86.1</td>
<td>86.4</td>
<td>86.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>86.6</td>
<td>86.5</td>
<td>86.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>86.6</td>
<td>86.5</td>
<td>86.5</td>
<td></td>
</tr>
</tbody>
</table>

**Radix Sort - 1M elements**

5x faster to sort 1M 64-bit ints

**Speedup**

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA A100</th>
<th>AMD MI250X</th>
</tr>
</thead>
<tbody>
<tr>
<td>coral</td>
<td>1.80x</td>
<td>1.25x</td>
</tr>
<tr>
<td>miniBUDE*</td>
<td>1.82x</td>
<td>1.92x</td>
</tr>
</tbody>
</table>
**Performance Portability of the Chapel Language on Heterogeneous Architectures**, 
Josh Milthorpe, Xianghao Wang, Ahmad Azizi (ORNL / ANU), HCW 2024

**GPU-Accelerated Tree-Search in Chapel: Comparing Against CUDA and HIP on Nvidia and AMD GPUs**, 
Guillaume Helbecque, Ezhilmathi Krishnasamy, Nouredine Melab, Pascal Bouvry (U. Luxembourg / U. Lille), PDCO 2024

---

**GPU Highlights Since CHIUW 2023: Community Papers at IPDPS Workshops**

Josh speaking today @ 9:40 PT

Guillaume speaking today @ 12:45 PT
**GPU Highlights Since CHIUW 2023: GPUs and User Codes**

- **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**
  - INRIA, IMEC, et al.

- **ChplUltra: Simulating Ultralight Dark Matter**
  - Nikhil Padmanabhan, J. Luna Zagorac, et al.
  - Yale University et al.

- **Arkouda**
  - Mike Merrill, Bill Reus, et al.
  - U.S. DoD

- **ChOp**
  - INRIA, IMEC, et al.

- **ChplUltra**
  - 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)**
  - Ben Albrecht et al.
  - Cray Inc. / HPE

- **CHGL: Chapel Hypergraph Library**
  - Louis Jenkins, Cliff Joslyn, Jesun Firoz, et al.
  - PNNL

**Active GPU efforts**
Investigating Portability in Chapel for Tree-based Optimization on GPU-powered Clusters

Tiago Carneiro, Engin Kayraklioglu, Guillaume Helbecque, Nouredine Melab (IMEC / HPE / U. Luxembourg, U. Lille),

Euro-Par 2024

Abstract. The Top500 list features supercomputers powered by accelerators from different vendors. This variety brings along with the heterogeneity challenge, both the code and performance portability challenges. In this context, Chapel’s native GPU support comes as a solution for code portability between different vendors. In this paper, we investigate the viability of using the Chapel high-productivity language as a tool to achieve both code and performance portability in large-scale tree-based search. As a case study, we implemented a distributed backtracking for solving permutation combinatorial problems. Extensive experiments conducted on big N-Queens problem instances, using up to 512 NVIDIA GPUs and 1024 AMD GPUs on Top500 supercomputers, reveal that it is possible to scale on the two different systems using the same tree-based search written in Chapel. This trade-off results in a performance decrease of less than 10% for the biggest problem instances.

GPU Highlights Since CHIUW 2023: Scaling on DOE Systems

Frontier (ORNL)
128 nodes x 8 GPUs = 1024 GPUs
GPUs: AMD Instinct MI250X

Perlmutter (NERSC)
128 nodes x 4 GPUs = 512 GPUs
GPUs: NVIDIA A100

Tiago speaking today @ 10:00 PT
also see Brett’s talk @ 10:10 PT
Additional GPU Resources

**GPU Programming in Chapel** blog series

- **Vendor-Neutral GPU Programming in Chapel**
- **July 31, Free and online**

---

**Chapel Language Blog**

About  Chapel Website  Featured  Series  Tags  Authors  All Posts

**GPU Programming in Chapel**

This series showcases Chapel's support for vendor-neutral GPU programming.

- **Introduction to GPU Programming in Chapel**
  Posted on January 10, 2024
  This post gives a beginner's introduction to Chapel's GPU programming features

- **Chapel's High-Level Support for CPU-GPU Data Transfers and Multi-GPU Programming**
  Posted on April 25, 2024
  This post covers how Chapel's arrays, parallelism, and locality features enable moving data between CPUs and GPUs.
GPU Highlights: ChapelCon 2024 Keynote

A Case for Parallel-First Languages in a Post-Serial, Accelerated World

Paul Sathre (Virginia Tech)

Abstract: Parallel processors have finally dominated all scales of computing hardware, from the personal and portable to the ivory tower datacenters of yore. However, dominant programming models and pedagogy haven't kept pace, and languish in a post-serial mix of libraries and language extensions. Further, heterogeneity in the form of GPUs has dominated the performance landscape of the last decade, penetrating casual user markets thanks to data science, crypto and AI booms. Unfortunately GPUs' performance remains largely constrained to expert users by the lack of more productive and portable programming abstractions. This talk will probe questions about how to rethink and democratize parallel programming for the masses. By reflecting on lessons learned from a decade and a half of accelerated computing, I'll show where Chapel 2.0 fits into the lineage of GPU computing, can capitalize on GPU momentum, and lead a path forward.

Bio: Paul Sathre is a Research Software Engineer in the Synergy Lab and NSF Center for Space, High-performance, and Resilient Computing (SHREC) at Virginia Tech. His research interests encompass systems software and tools and programming systems, particularly with respect to democratizing access to high-performance computing. More recently, his work has focused on the intersection of computational co-design with portable and productive languages, tools, and libraries for heterogeneous computing.
Scalability and Performance
Bale IndexGather in Chapel vs. SHMEM on HPE Cray EX

Bale IndexGather Performance

HPE Cray EX (Slingshot-11)

Number of Locales

GB/s
Since CHIUW 2023, we have evaluated performance of various communication patterns x networks

- For example, ISx on Slingshot-11, InfiniBand, and AWS:

  **Network:** Slingshot-11 (single NIC)
  **Processor:** Dual-socket AMD Milan

  **Network:** InfiniBand HDR-200 (single NIC)
  **Processor:** Dual-socket AMD Milan

  **Network:** AWS (Ethernet and EFA)
  **Processor:** AWS Graviton3

Notes:
- These AWS results should be considered preliminary and have almost certainly improved since this study
- We also looked briefly at RDMA-enabled Ethernet (not shown here)
Chapel Co-Locales

- A **locale** in Chapel is a part of the target system that can run tasks and store variables
  - Traditionally, each compute node has been a locale

- We’ve recently added support for multiple **co-locales** per compute node:
  - Command-line interface: `$./myChplProg -nl 8x2 # use 8 nodes w/ 2 locales each...
  - Typical cases:
    - locale per NIC
    - locale per socket
    - locale per NUMA domain
    - locale per L3 cache

- Co-locales can improve performance via:
  - better network utilization
  - better memory locality and affinity

<table>
<thead>
<tr>
<th>Configuration</th>
<th>GB/s</th>
<th>Improvement</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>-nl 2</td>
<td>357</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>-nl 2x2</td>
<td>460</td>
<td>28.9%</td>
<td>Socket</td>
</tr>
<tr>
<td>-nl 2x8</td>
<td>466</td>
<td>30.5%</td>
<td>NUMA</td>
</tr>
<tr>
<td>-nl 2x16</td>
<td>470</td>
<td>31.7%</td>
<td>L3 cache</td>
</tr>
<tr>
<td>“first touch”</td>
<td>470</td>
<td>31.7%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Stream TRIAD on dual-socket node, Milan CPUs, 64 cores/CPU
We’ve also improved serial/vector performance this year

- In 2023, our fastest CLBG n-body lagged the baseline by 1.4x
  - Rust, Julia, Fortran, and C++ versions all outperformed it

- This year, we became the baseline after the 2.0 release
  - With no source changes!

I’ll speak briefly on the CLBG today @ 9:30 PT
Tool Improvements
Chapel Tools: Background

- Tools have been a classic chicken-and-egg problem for Chapel
  - Users don’t want to use a new language without tools
  - Tools teams don’t want to support a new language until it has users
  - Meanwhile, our team has traditionally had trouble prioritizing tools work

- What tools we have typically come from the open-source community

Happily, this has started to change through our Dyno compiler re-work effort

see Henry, Drake, and Cole’s talk @ 8:35 PT
Three New Tools Since CHIUW 2023

**chpl-language-server (CLS):**

- Enables features within editors that support the Language Server Protocol (LSP) — VSCode, vim, emacs, ...
- Provides real-time features to navigate, query, and refactor Chapel code

**chplcheck:**

- A Chapel linter that provides style checks and helps prevent common errors
- Can be run from the command-line or an editor (via LSP)

**VSCode extension for Chapel:**

- Supports the two tools above, along with syntax highlighting, autofill, GUI breakpoints, ...

see Daniel and Jade's talk @ 8:55 PT
Time’s running short...
Yet there’s so much I could cover!
There Are Many Other Exciting Talks Today (that I couldn’t weave into this talk)

12:20–12:30 Exploring Machine Learning Capabilities in Chapel: An Internship Journey
Iain Moncrief (Oregon State University)

Abstract: This talk recounts a recent internship experience with learning Chapel.

12:55–1:15 Arrays as Arguments in First-Class Functions: the Levenberg-Marquardt Algorithm in Chapel
Nelson Dias, Débora Roberti and Vanessa Arruda Dias (Federal University of Paraná, Federal University of Santa Maria)

1:15–1:35 On the Design of Graph Analytical Software in Chapel
Oliver Alvarado Rodriguez, David A. Bader and Zhihui Du (New Jersey Institute of Technology)

1:35–1:45 Implementing Imaginary Elementary Mathematical Functions
Damian McGuckin, Peter Harding (Pacific ESI)

2:00–2:10 Chapel in a Petabyte-Scale GPU Database Engine with Voltron Data’s Theseus
Trent Nelson and Fernanda Foerther (Voltron Data)

2:10–2:20 Chplx: an HPX Foundation for Chapel
Shreyas Atre, Chris Taylor, Patrick Diehl and Hartmut Kaiser (Louisiana State University, Tactical Computing Labs, LLC)

2:20–2:40 Follow-Up on Chapel-Powered HPC Workflows for Python
John Byrne, Harumi Kuno, Chinmay Ghosh, Pornro Shome, Amitha C, Sharad Singhal, Clarete Riana Crasta, David Emberson and Abhishek Dwaraki (Hewlett Packard Enterprise)

Iain speaking today @ 12:20 PT
Nelson speaking today @ 12:55 PT
Oliver speaking today @ 1:15 PT
Damian speaking today @ 1:35 PT
Trent speaking today @ 2:00 PT
Chris speaking today @ 2:10 PT
Harumi speaking today @ 2:20 PT
Updates from other CHIUW Alumni

CHAMPs: 3D Unstructured CFD

Arkouda: Interactive Data Science at Massive Scale

ChOp: Chapel-based Optimization

ChplUltra: Simulating Ultralight Dark Matter

Lattice-Symmetries: a Quantum Many-Body Toolbox


RapidQ: Mapping Coral Biodiversity

ChapQG: Layered Quasigeostrophic CFD

Chapel-based Hydrological Model Calibration

CrayAI HyperParameter Optimization (HPO)

CHGL: Chapel Hypergraph Library

? (images provided by their respective teams and used with permission)
Update from the CHAMPS Team

CHAMPS status update

- CHAMPS has seen very rapid development progress since its initiation, in many ways due to the efficiency of Chapel. Alongside the technical developments of our Computational Fluid Dynamics (CFD) capabilities, scientific progress has been very fruitful.

- **The research group is now into more holistic research**, slowing down production quantity to examine more fundamental, high-impact research of similar high quality but with computational breath that is one-order of magnitude more complex. For instance:
  - **problem size has now reached 2 Billion unknowns**
  - Reynolds-Averaged Navier-Stokes has moved from steady to unsteady flow analysis, requiring 1000x more calculations/solutions.

- This is due to the industrial pull, which requires very advanced computational workflows to examine **novel aircraft configurations to attain environmental targets** such as Bombardier Eco-Jet or NASA/Boeing Truss-Braced Wing concepts.

- **Important advances will be presented at the American Institute of Aeronautics and Astronautics AVIATION conference in late July 2024. Stay tuned!**
Updates from other CHIUW Alumni

CHAMPS: 3D Unstructured CFD

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

ChplUltra: Simulating Ultralight Dark Matter

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

Arkouda: Interactive Data Science at Massive Scale

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

ChOp: Chapel-based Optimization

INRIA, IMEC, et al.

Arkouda: Interactive Data Science at Massive Scale

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

ChOp: Chapel-based Optimization

INRIA, IMEC, et al.

Lattice-Symmetries: a Quantum Many-Body Toolbox


RapidQ: Mapping Coral Biodiversity

ChapQG: Layered Quasigeostrophic CFD

Chapel-based Hydrological Model Calibration

CrayAI HyperParameter Optimization (HPO)

CHGL: Chapel Hypergraph Library

Your Application Here?

(images provided by their respective teams and used with permission)
Publications at PAW-ATM 2023 (an SC23 workshop)

High-Performance Programming and Execution of a Coral Biodiversity Mapping Algorithm Using Chapel, Scott Bachman (NCAR) et al.

Implementing Scalable Matrix-Vector Products for the Exact Diagonalization Methods in Quantum Many-Body Physics, Tom Westerhout (Radboud University) et al.

Coral Reef Biodiversity (image analysis)
Performance

Previous performance (serial, MATLAB): ~ Multiple days
Current performance (360x cores, Chapel): ~ 2 seconds
Roughly 5 orders of magnitude improvement

Implementing Scalable Matrix-Vector Products for the Exact Diagonalization Methods in Quantum Many-Body Physics

Tom Westerhout
tom.westerhout@ru.nl
Institute for Molecules and Materials, Radboud University
Nijmegen, The Netherlands

Bradford L. Chamberlain
bradford.chamberlain@hp.com
Hewlett Packard Enterprise

ABSTRACT

Exact diagonalization is a well-established method for simulating small quantum systems. Its applicability is limited by the exponential growth of the Hamiltonian matrix that needs to be diagonalized. Physical symmetries are usually utilized to reduce the matrix dimension, and distributed-memory parallelism is employed to exploit larger systems. This paper focuses on an implementation of the exact diagonalization methods, with a special emphasis on the matrix-vector product. Instead of the conventional MPI-X paradigm, Chapel is chosen as the language in this work.

We provide a comprehensive description of the algorithms and present performance and scalability tests. Our implementation outperforms the state-of-the-art MPI-based solution by a factor of 7–8 on 32 compute nodes or 496 cores and scales well through 238 nodes or 32,384 cores. The implementation has 3 times fewer software lines of code than the current state-of-the-art, but is still able to handle generic Hamiltonians.

CCS CONCEPTS
- Computing methodologies → Parallel computing methodologies
- Applied computing → Physics
- Theory of computation → Distributed algorithms
Consider submitting work on applications in Chapel to:

The 7th Annual Parallel Applications Workshop, Alternatives To MPI+X

November 17 | 18, 2024
Held in conjunction with SC24

Deadline: July 24, 2024
Submission Styles: Papers / Talks
Updates from other CHIUW Alumni

CHAMS: 3D Unstructured CFD
Arkouda: Interactive Data Science at Massive Scale

CHIUW 2021  CHIUW 2022

CHIUW 2020  CHIUW 2023

CHOp: Chapel-based Optimization
ChplUltra: Simulating Ultralight Dark Matter

CHIUW 2021  CHIUW 2023

CHIUW 2020  CHIUW 2022

Lattice-Symmetries: a Quantum Many-Body Toolbox

ChIUW 2022

RapidQ: Mapping Coral Biodiversity
ChapQG: Layered Quasigeostrophic CFD

CHIUW 2023

Chapel-based Hydrological Model Calibration
CrayAI HyperParameter Optimization (HPO)

CHIUW 2023

CHGL: Chapel Hypergraph Library

CHIUW 2020

(images provided by their respective teams and used with permission)
Newly Minted PhDs

• Since her CHIUW 2023 talk, **Marjan Asgari**...
  • Defended her Ph.D. on parallel computing for calibrating hydrological models
    – Her software is still in use by her former research group at University of Guelph
  • Now works for **Natural Resources Canada** focused on parallel computing

• Since his CHIUW 2023 talk, **Thomas Rolinger**...
  • Defended his Ph.D. on compiler optimizations for irregular memory access patterns in PGAS languages
    – Published a paper on his work at IPDPS 2024
  • Now works on **NVIDIA**’s back-end compiler team

**Congratulations, Marjan and Thomas!!!**
Wrapping Up
The Chapel Team at HPE
State of the Chapel Project: Summary

As a team and community, we have accomplished a ton since CHIUW 2023

- Released Chapel 2.0 and made ongoing improvements
- Accomplished and published great scientific results
- Improved GPU, tool, and performance features
- Got a great start on renewed focus on community

Stay tuned to today’s talks to hear more detail about many of these efforts!

We look forward to working with you all in the year to come!
Chapel Resources

**Chapel homepage:** [https://chapel-lang.org](https://chapel-lang.org)
- (points to all other resources)

**Blog:** [https://chapel-lang.org/blog/](https://chapel-lang.org/blog/)

**Social Media:**
- Facebook: [@ChapelLanguage](https://www.facebook.com/ChapelLanguage)
- LinkedIn: [@chapel-programming-language](https://www.linkedin.com/company/chapel)
- Mastadon: [@ChapelProgrammingLanguage](https://mastodon.social/@ChapelProgrammingLanguage)
- X / Twitter: [@ChapelLanguage](https://twitter.com/ChapelLanguage)
- YouTube: [@ChapelLanguage](https://www.youtube.com/@ChapelLanguage)

**Community Discussion / Support:**
- Discourse: [https://chapel.discourse.group/](https://chapel.discourse.group/)
- Gitter: [https://gitter.im/chapel-lang/chapel](https://gitter.im/chapel-lang/chapel)
- Stack Overflow: [https://stackoverflow.com/questions/tagged/chapel](https://stackoverflow.com/questions/tagged/chapel)
- GitHub Issues: [https://github.com/chapel-lang/chapel/issues](https://github.com/chapel-lang/chapel/issues)
Consider submitting work on applications in Chapel to:

The 7th Annual Parallel Applications Workshop, Alternatives To MPI+X

November 17 | 18, 2024
Held in conjunction with SC24

Deadline: July 24, 2024
Submission Styles: Papers / Talks
Thank you

https://chapel-lang.org
@ChapelLanguage