



**Hewlett Packard
Enterprise**

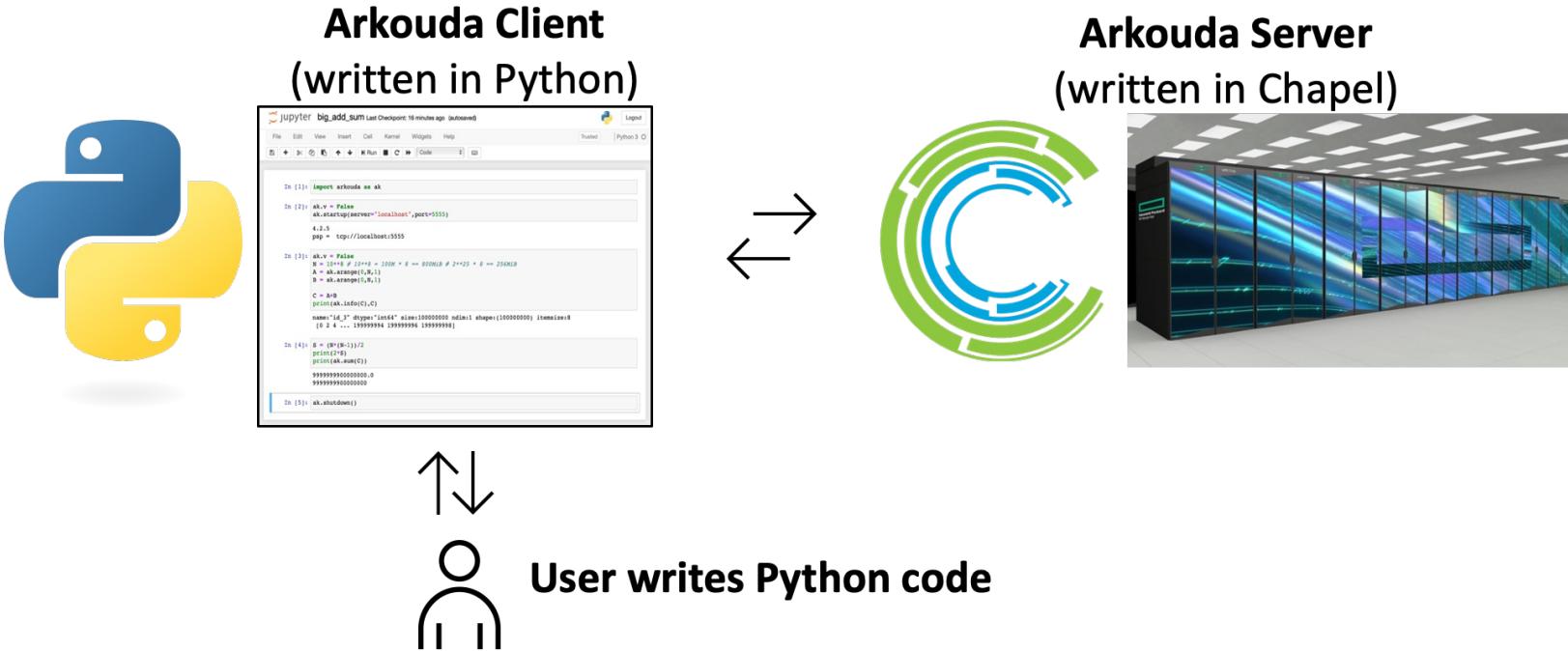
Arkouda and Chapel: Highlights Since CLSAC 2022

Brad Chamberlain
November 6, 2024



What is Arkouda?

Q: “What is Arkouda?”



What is Chapel?

Chapel: A modern parallel programming language

- Pythonic readability, writability, maintainability
- Performs/scales like Fortran, C/C++, MPI, SHMEM, OpenMP, CUDA, ...
- Safety features w.r.t. types, parallelism, memory



Bale IndexGather in Chapel vs. SHMEM on HPE Cray EX (Slingshot-11)

Chapel (Simple / Auto-Aggregated version)

```
forall (d, i) in zip(Dst, Inds) do
    d = Src[i];
```

Chapel (Explicitly Aggregated version)

```
forall (d, i) in zip(Dst, Inds) with
    (var agg = new SrcAggregator(int)) do
        agg.copy(d, Src[i]);
```

SHMEM (Exstack version)

```
i=0;
while( exstack_proceed(ex, (i==l_num_req)) ) {
    i0 = i;
    while(i < l_num_req) {
        l_idx = pckindx[i] >> 16;
        pe = pckindx[i] & 0xffff;
        if(!exstack_push(ex, &l_idx, pe))
            break;
        i++;
    }

    exstack_exchange(ex);

    while(exstack_pop(ex, &idx , &fromth)) {
        idx = ltable[idx];
        exstack_push(ex, &idx, fromth);
    }
    lgp_barrier();
    exstack_exchange(ex);

    for(j=i0; j<i; j++) {
        fromth = pckindx[j] & 0xffff;
        exstack_pop_thread(ex, &idx, (uint64_t)fromth);
        tgt[j] = idx;
    }
    lgp_barrier();
}
```

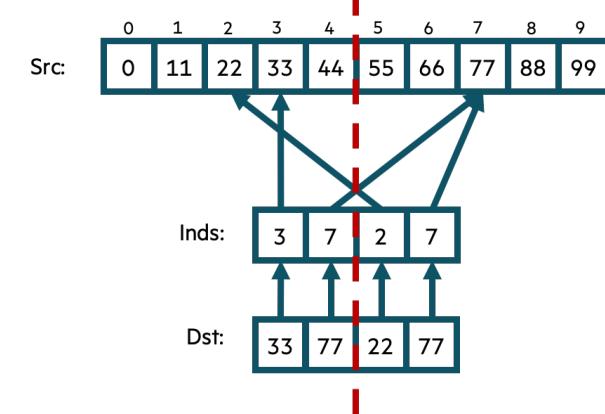
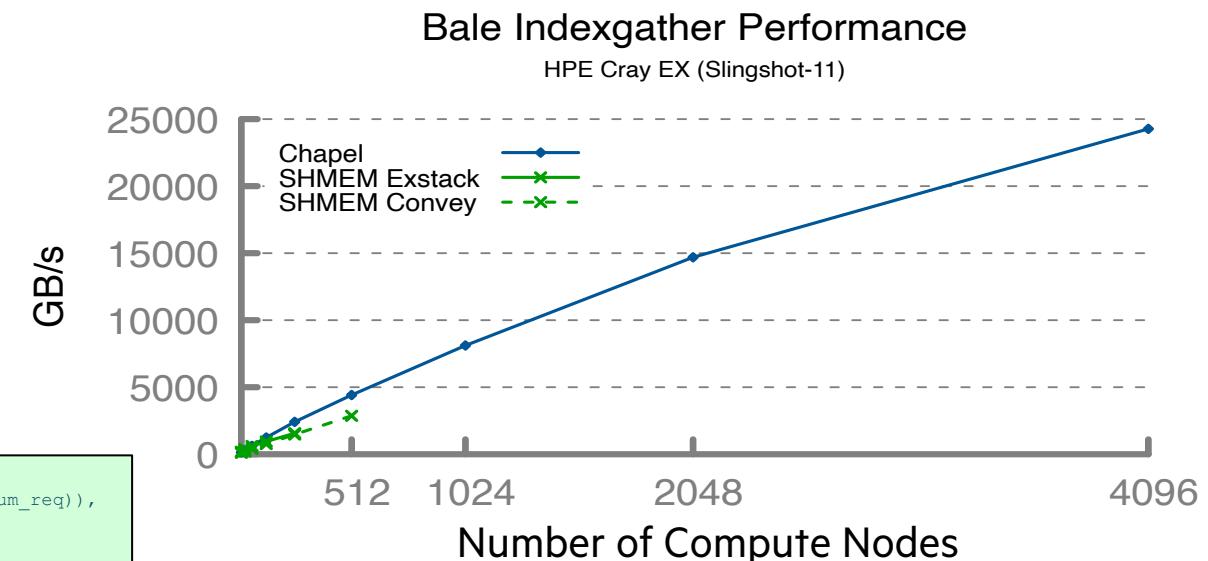
SHMEM (Conveyors version)

```
i = 0;
while (more = convey_advance(requests, (i == l_num_req)),
       more | convey_advance(replies, !more)) {

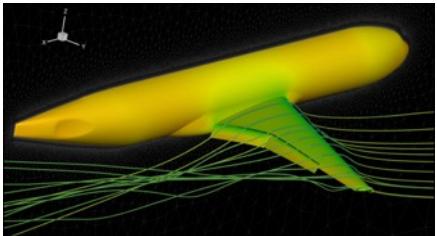
    for (; i < l_num_req; i++) {
        pkg.idx = i;
        pkg.val = pckindx[i] >> 16;
        pe = pckindx[i] & 0xffff;
        if (!convey_push(requests, &pkg, pe))
            break;
    }

    while (convey_pull(requests, ptr, &from) == convey_OK) {
        pkg.idx = ptr->idx;
        pkg.val = ltable[ptr->val];
        if (!convey_push(replies, &pkg, from)) {
            convey_unpull(requests);
            break;
        }
    }

    while (convey_pull(replies, ptr, NULL) == convey_OK)
        tgt[ptr->idx] = ptr->val;
}
```

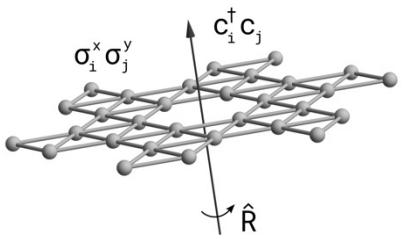


Applications of Chapel



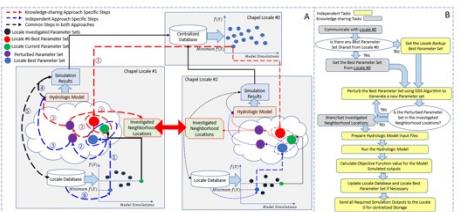
CHAMPS: 3D Unstructured CFD

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



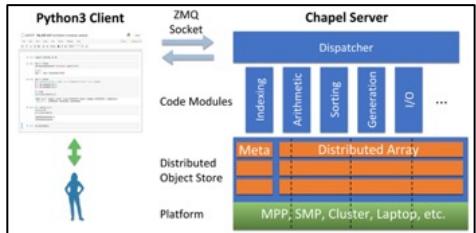
Lattice-Symmetries: a Quantum Many-Body Toolbox

Tom Westerhout
Radboud University



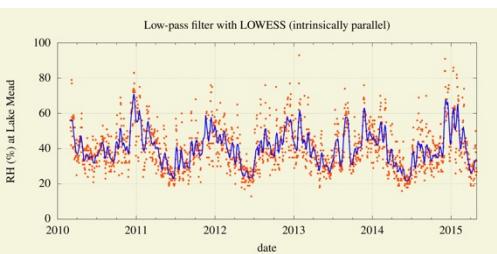
Chapel-based Hydrological Model Calibration

Marjan Asgari et al.
University of Guelph



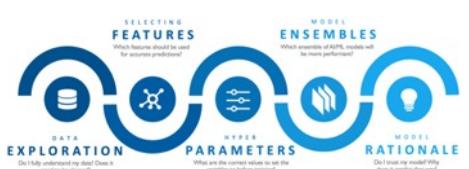
Arkouda: Interactive Data Science at Massive Scale

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



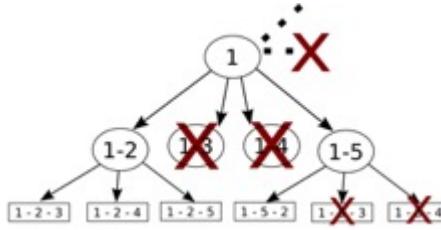
Desk dot chpl: Utilities for Environmental Eng.

Nelson Luis Dias
The Federal University of Paraná, Brazil



CrayAI HyperParameter Optimization (HPO)

Ben Albrecht et al.
Cray Inc. / HPE



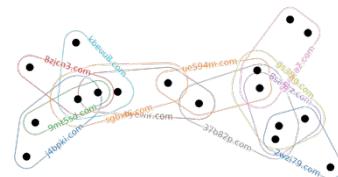
ChOp: Chapel-based Optimization

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



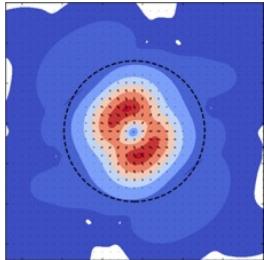
RapidQ: Mapping Coral Biodiversity

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



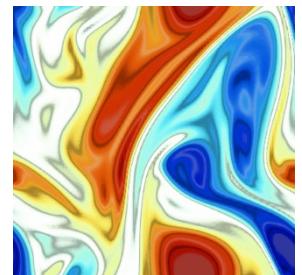
CHGL: Chapel Hypergraph Library

Louis Jenkins, Cliff Joslyn, Jesun Firoz, et al.
PNNL



ChplUltra: Simulating Ultralight Dark Matter

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



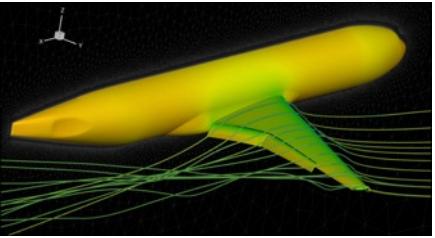
ChapQG: Layered Quasigeostrophic CFD

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



Your Application Here?

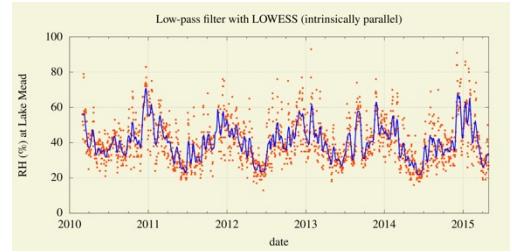
Chapel's users reap its benefits at varying scales of systems and code sizes



Computation: aircraft simulation
Code size: 100,000+ lines
Systems: desktops, HPC systems



Computation: coral reef image analysis
Code size: ~300 lines
Systems: desktops, HPC systems w/ GPUs



Computation: ATTO data analysis
Code size: 5000+ lines
Systems: desktops w/ CPUs & GPUs (only)



Chapel Blog Series: 7 Questions for Chapel Users



About Chapel Website Featured Series Tags Authors All Posts



7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel

Posted on October 15, 2024.

Tags:

By: Engin Kayraklıoğlu

This is the first in a new series of questions about to shine a light on ways in which you are using Chapel and someone who is and ought to be.

For our inaugural edition of one of Chapel's most ambitious users, we turn to Éric and CHAMPS, but he

1. Who are you?

My name is Éric Laurendeau, my Bachelor's in Canada (I



Chapel Language Blog

About Chapel Website Featured Series Tags Authors All Posts

7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel

Posted on October 15, 2024.

Tags:

By: Brad Chamberlain



Chapel Language Blog

About Chapel Website Featured Series Tags Authors All Posts

7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel

Posted on October 15, 2024.

Tags:

User Experiences

Interviews

Data Analysis

Computational Fluid Dynamics

By: Engin Kayraklıoğlu, Brad Chamberlain



In this second installment of our Seven Questions for Chapel Users series, we turn to Dr. Nelson Luis Dias from Brazil who is using Chapel to analyze data generated by the Amazon Tall Tower Observatory (ATTO), a project dedicated to long-term, 24/7 monitoring of greenhouse gas fluctuations. Read on to learn more about his work and use of Chapel!

This interview was conducted live (with assistance).

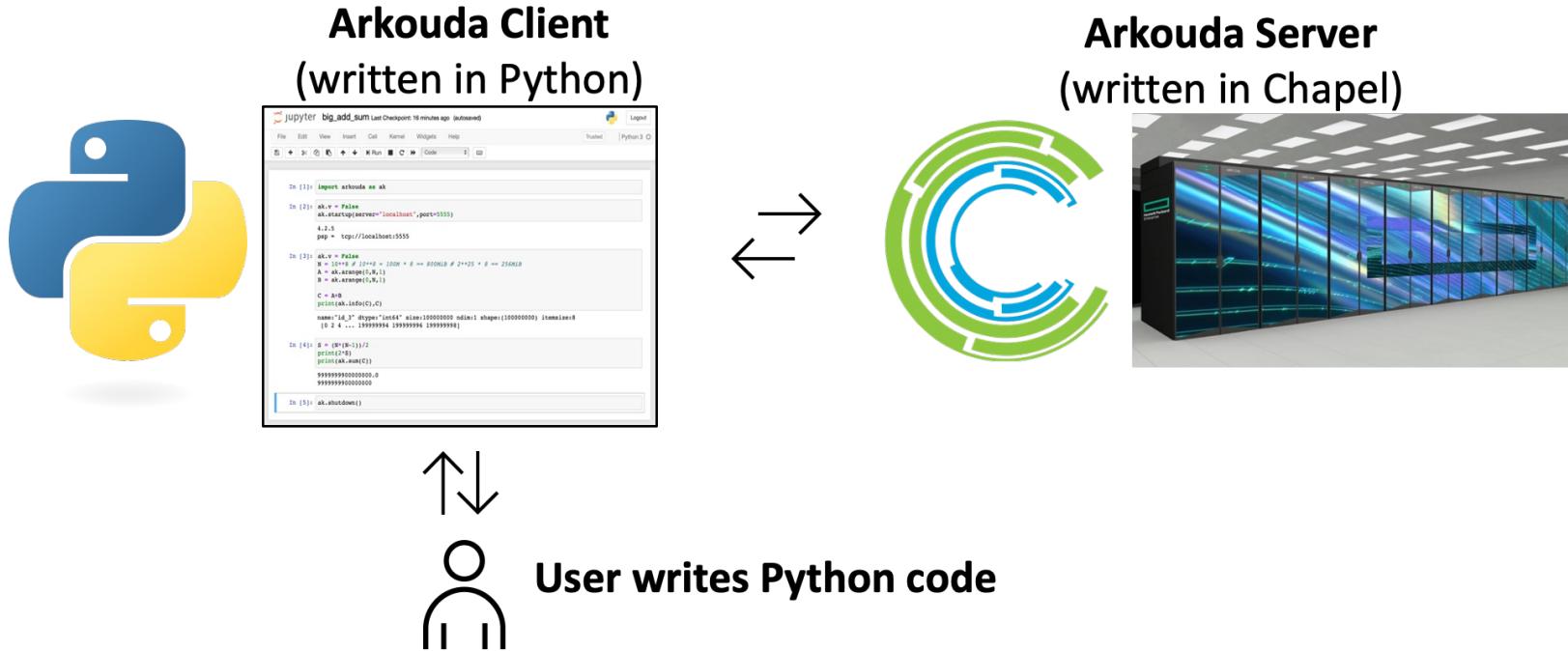


available at: <https://chapel-lang.org/blog/series/7-questions-for-chapel-users/>



What is Arkouda?

Q: “What is Arkouda?”



A1: “A scalable version of NumPy / Pandas for data scientists”

A2: “An extensible framework for arbitrary HPC computations”

A3: “A way to drive HPC systems interactively from Python on a laptop”

In Memoriam

- Mike Merrill passed away two years ago this Friday
- 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





Hewlett Packard
Enterprise

Arkouda and Chapel: Updates I'd want Mike to know



Brad Chamberlain
November 6, 2024

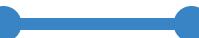
Arkouda Argsort Scalability

HPE Cray EX (May 2023)



- Slingshot-11 network (200 Gb/s)
- 8192 compute nodes
- 256 TiB of 8-byte values
- ~8500 GiB/s (~31 seconds)

HPE Cray EX (April 2023)

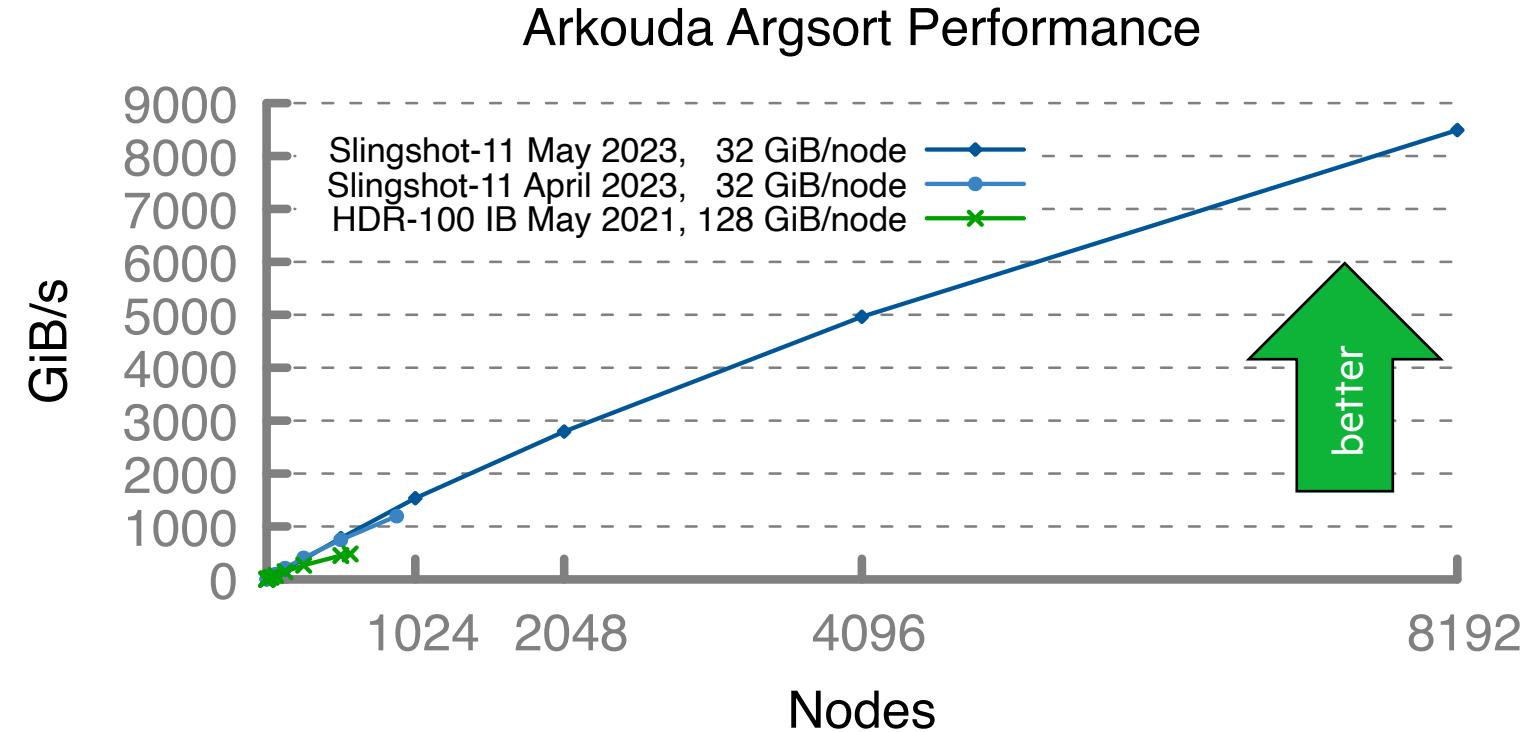


- Slingshot-11 network (200 Gb/s)
- 896 compute nodes
- 28 TiB of 8-byte values
- ~1200 GiB/s (~24 seconds)

HPE Apollo (May 2021)



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



Achieved using ~100 lines of Chapel

Arkouda's Modularity Is Being Leveraged

- Arkouda servers can be built by mixing and matching various modules
 - Initial modules supported NumPy and Pandas
- Key examples include:
 - Graph Processing
 - Multidimensional Arrays supporting the Python Array API
 - Sparse Matrix Computations

**Jeremiah Corrado will demo this for
the Pangeo community, Nov 20th**



New Tooling that Simplifies Adding New Arkouda Operations

Was:

```
/* intIndex "a[int]" response to __getitem__(int) */
@arkouda.registerND(cmd_prefix="[int]")
proc intIndexMsg(cmd: string, msgArgs: borrowed MessageArgs, st: borrowed SymTab, param nd: int): MsgTuple throws {
    param pn = Reflection.getRoutineName();
    var repMsg: string; // response message
    var idx = msgArgs.get("idx").getTuple(nd);
    const name = msgArgs.getValueOf("array");
    imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),
        "%s %s %?".format(cmd, name, idx));
    var gEnt: borrowed GenSymEntry = getGenericTypedArrayEntry(name, st);

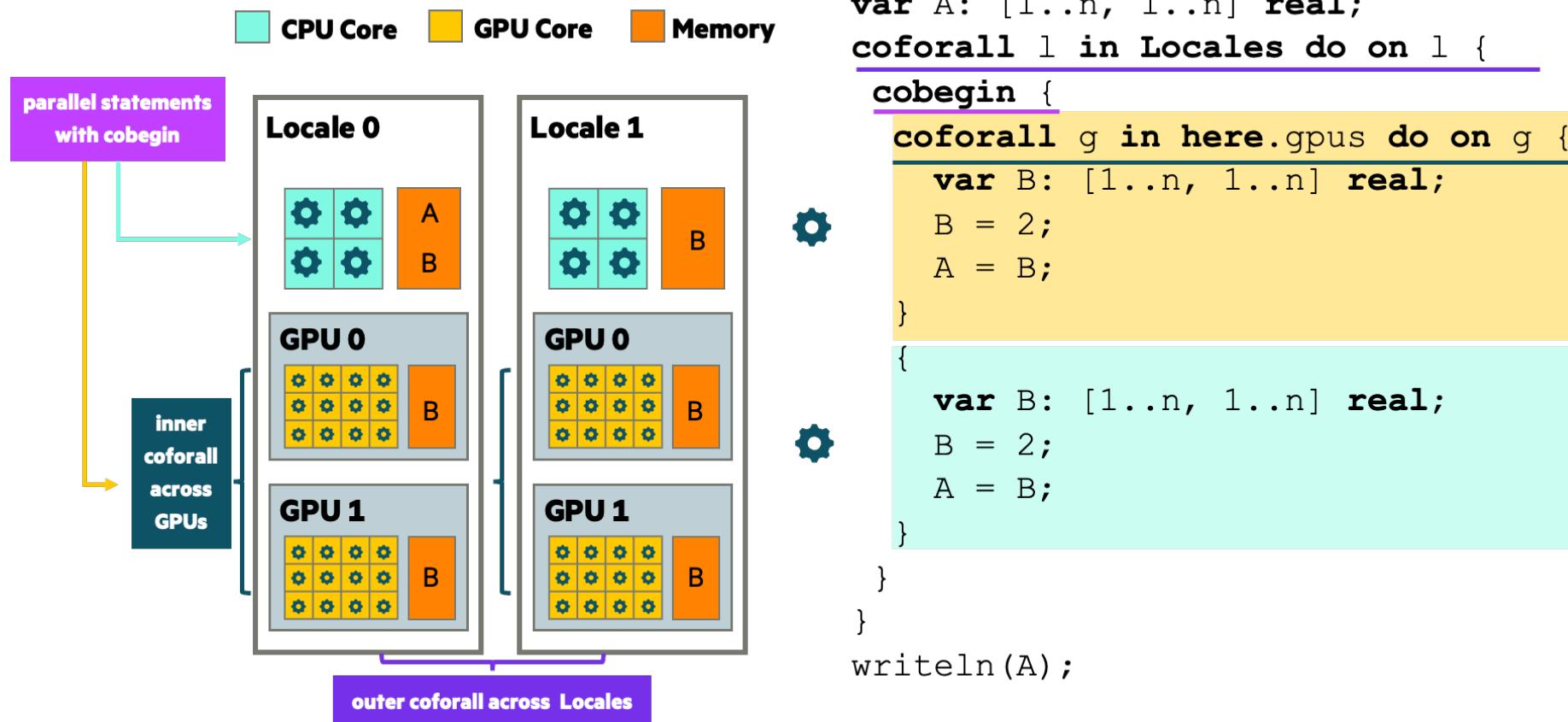
    select (gEnt.dtype) {
        when (DType.Int64) {
            var e = toSymEntry(gEnt, int, nd);
            repMsg = "item %s %?".format(dtype2str(e.dtype),e.a[...idx]);
            imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),repMsg);
            return new MsgTuple(repMsg, MsgType.NORMAL);
        }
        when (DType.UInt64) {
            var e = toSymEntry(gEnt, uint, nd);
            repMsg = "item %s %?".format(dtype2str(e.dtype),e.a[...idx]);
            imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),repMsg);
            return new MsgTuple(repMsg, MsgType.NORMAL);
        }
        when (DType.Float64) {
            var e = toSymEntry(gEnt, real, nd);
            repMsg = "item %s %.17r".format(dtype2str(e.dtype),e.a[...idx]);
            imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),repMsg);
            return new MsgTuple(repMsg, MsgType.NORMAL);
        }
        when (DType.Bool) {
            var e = toSymEntry(gEnt,bool, nd);
            repMsg = "item %s %?".format(dtype2str(e.dtype),e.a[...idx]);
            repMsg = repMsg.replace("true","True"); // chapel to python bool
            repMsg = repMsg.replace("false","False"); // chapel to python bool
            imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),repMsg);
            return new MsgTuple(repMsg, MsgType.NORMAL);
        }
        when (DType.BigInt) {
            var e = toSymEntry(gEnt,bigint, nd);
            repMsg = "item %s %?".format(dtype2str(e.dtype),e.a[...idx]);
            imLogger.debug(getModuleName(),getRoutineName(),getLineNumber(),repMsg);
            return new MsgTuple(repMsg, MsgType.NORMAL);
        }
        otherwise {
            var errorMsg = notImplementedError(pn,dtype2str(gEnt.dtype));
            imLogger.error(getModuleName(),getRoutineName(),getLineNumber(),errorMsg);
            return new MsgTuple(errorMsg, MsgType.ERROR);
        }
    }
}
```

Now:

```
@arkouda.registerCommand("[int]")
proc intIndex(const ref array: [?d] ?t, idx: d.rank*int): t {
    return array[idx];
}
```

Chapel and GPUs

- Chapel now supports vendor-neutral GPU programming (NVIDIA and AMD)
 - Uses the identical features as for programming multi-core CPUs and HPCs



GPU Highlights: Paul Sathre's ChapelCon 2024 Keynote



A case for parallel-first languages in
a post-serial, accelerated world

Paul Sathre
Research Software Engineer
Synergy Lab & NSF Center for Space, High-Performance and Resilient Computing
Virginia Tech



Intro

What I really care about:

Closing the gaps between the
parallel hardware we already have,
and **the people** who could benefit from it

So how do we **enable** them?
(Conversely, what are the **barriers** to use?)

4



Sathre, P. "A case for parallel-first languages in a post-serial, accelerated world"
ChapelCon'24 -- June 7, 2024



Act I: Parallelism is everywhere ...

Post-serial or “serial with sprinkles”

- Dominant programming models are still **post-serial**
 - “Sprinkles”: **optional** libraries, pragmas, language extensions
- Chapel presents a different option: **parallel-first**
 - A non-separable part of the **keywords**, **data abstractions**, and **semantics** of the language (i.e. **promotion**)



15

ChAI: Chapel AI

- A native PyTorch-esque Chapel module
 - Supports tensors, training, and inference
 - Runs locally, distributed, and/or on GPUs
 - Can be integrated into other HPC codes
 - e.g., Arkouda
- Implemented by Iain Moncrief
 - junior at Oregon State University
 - notably, written in one summer internship

The screenshot shows the GitHub repository page for 'ChAI' (Chapel Artificial Intelligence). The repository is public and maintained by 'lainmon'. It has 5 branches and 1 tag. The main branch contains 329 commits from 'lainmon'. Recent commits include updates to documentation, examples, and the presentation folder. The repository also includes scripts, a src folder with tests, and a test folder. A .gitignore file is present. The README file describes ChAI as a library for AI/ML in Chapel, highlighting its parallel nature and suitability for AI/ML tasks.

Code | **Issues 7** | **Pull requests 2** | **Actions** | **Projects** | **Security** | **Insights**

Code | **Code**

About

A Chapel library for Machine Learning that supports distributed inference, automatic differentiation, and CUDA/HIP utilization.

iainmon.github.io/ChAI

Readme | **Activity** | **3 stars** | **3 watching** | **4 forks** | Report repository

Releases 1

[Intern Presentation Snapshot](#) [Latest] on Aug 31

Packages

No packages published

Contributors 3

lainmon Iain Moncrief
 jeremiah-corrado
 DanilaFe Daniel

ChAI: Chapel Artificial Intelligence

ChAI is a library for AI/ML in [Chapel](#). Due to Chapel's highly parallel nature, it is well-suited for AI/ML tasks; the goal of the library is to provide a foundation for such tasks, enabling local, distributed, and GPU-enabled computations.

But wait, there's more!

- **AWS/EFA:** Now supported by Chapel and Arkouda
- **Spack/E4S:** Now support Chapel (Arkouda in-progress)
- **Chapel 2.0:** released this year, providing forward-compatibility
- **Parquet:** improved and optimized support
- **New websites:** Arkouda's is online, Chapel's will launch soon
- ...



The Arkouda website homepage features a dark background with a white central area. At the top, there's a logo of a koala and navigation links for "github", "documentation", and "gitter". The main headline reads "Massive-scale data science, from the comfort of your laptop". Below this, there are two sections: "Arkouda Ready for supercomputers" and "NumPy Industry standard". A code snippet shows how to launch an Arkouda server and perform basic operations like generating large arrays and sorting them. At the bottom, there are three buttons: "Try it Out" (pink), "Tutorial Video" (white), and "Chat on Gitter" (white).

The Chapel Programming Language website has a light blue header with navigation links for "DOWNLOAD", "DOCS", "LEARN", "RESOURCES", "COMMUNITY", and "BLOG". The main title is "The Chapel Programming Language" with the subtitle "Productive parallel computing at every scale". On the left, there's a sidebar with a "Hello World" example and other options like "Distributed Hello World", "Parallel File IO", "1D Heat Diffusion", and "GPU Kernel". In the center, there's a code editor showing a parallel task example. Below the editor, there are three buttons: "TRY CHAPEL", "GET CHAPEL", and "LEARN CHAPEL". The bottom section is divided into three panels: "PRODUCTIVE" (concise and readable), "PARALLEL" (built from the ground up for parallel algorithms), and "FAST" (generates efficient machine code). A small note in the "FAST" panel states: "Chapel is a compiled language, generating efficient machine code that meets or beats the performance of other languages."

Thank you

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

