Hewlett Packard
Enterprise

Arkouda and Chapel: Highlights Since CLSAC 2022

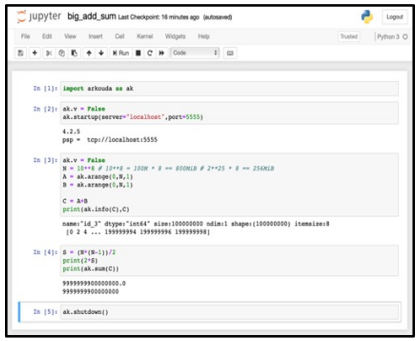
Brad Chamberlain

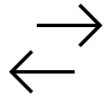
November 6, 2024

What is Arkouda?

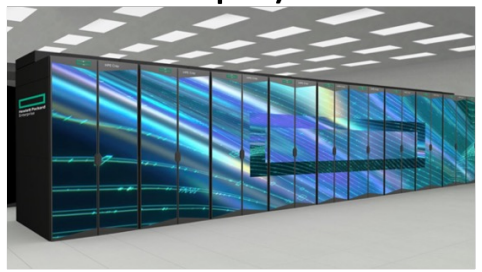
Q: “What is Arkouda?”

Arkouda Client
(written in Python)





Arkouda Server
(written in Chapel)



User writes Python code

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)

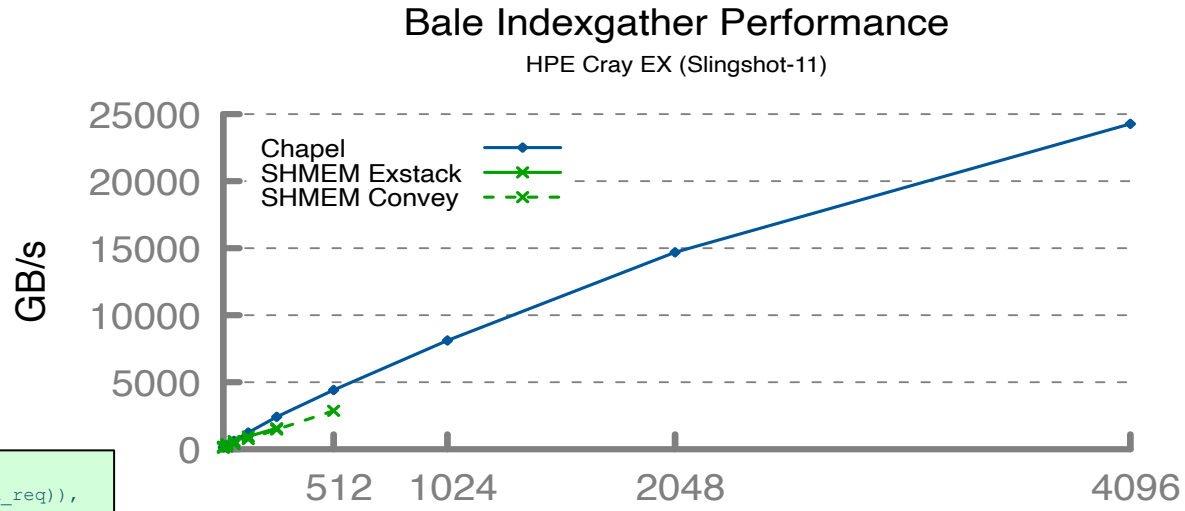
```
forall (d, i) in zip(Dst, Inds) with
 (var agg = new SrcAggregator(int)) do
  agg.copy(d, Src[i]);
```

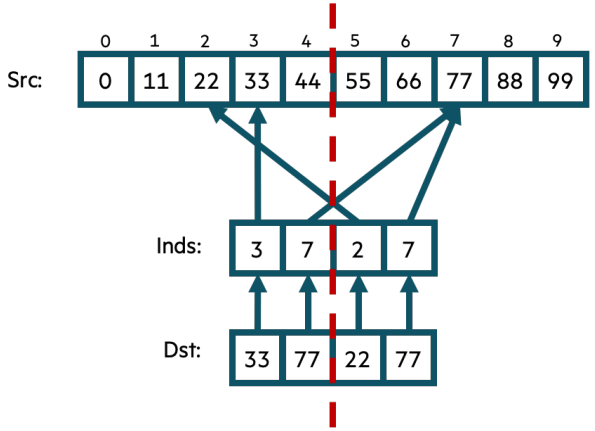
SHMEM (Exstack version)

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

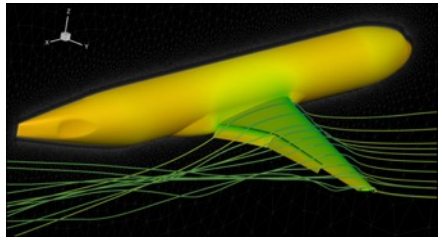
SHMEM (Conveyors version)

```
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;
}
```

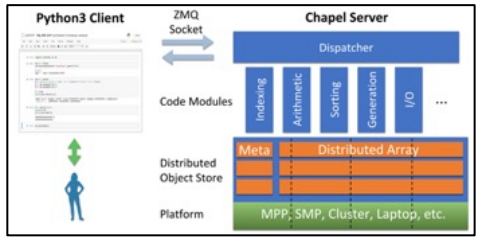




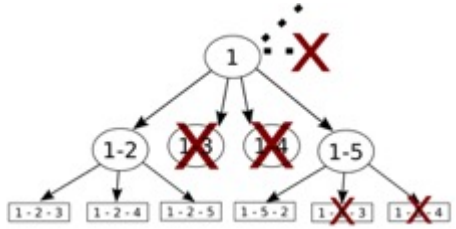
Applications of Chapel



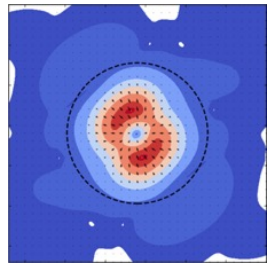
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



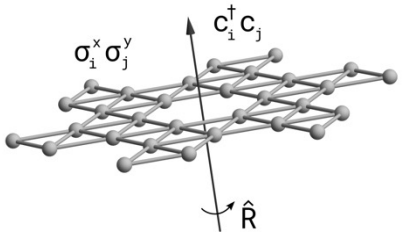
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



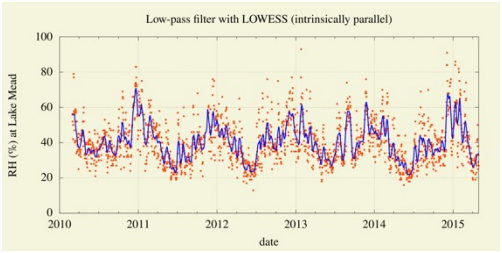
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

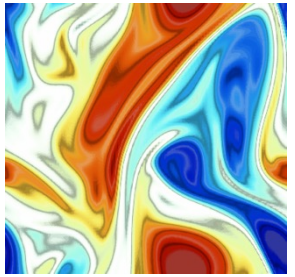


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

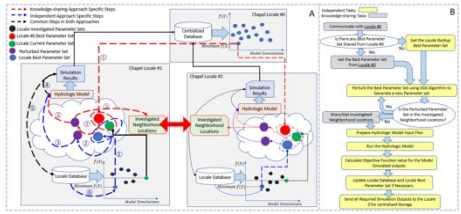


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

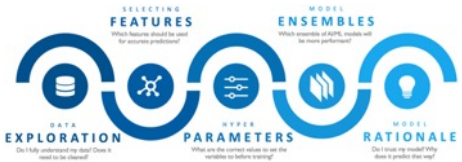
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



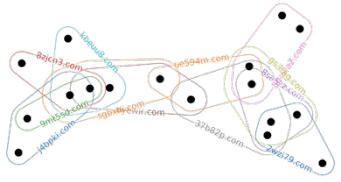
ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.



Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE



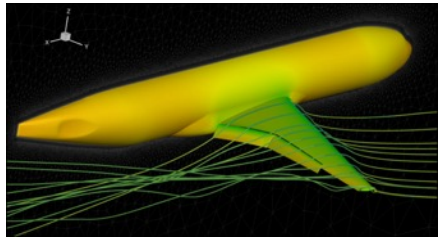
CHGL: Chapel Hypergraph Library
Louis Jenkins, Cliff Joslyn, Jesun Firoz, et al.
PNNL

?

Your Application Here?

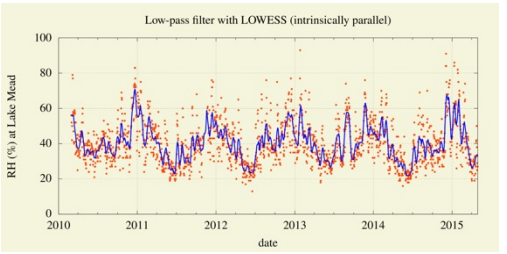
(images provided by their respective teams and used with permission)

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



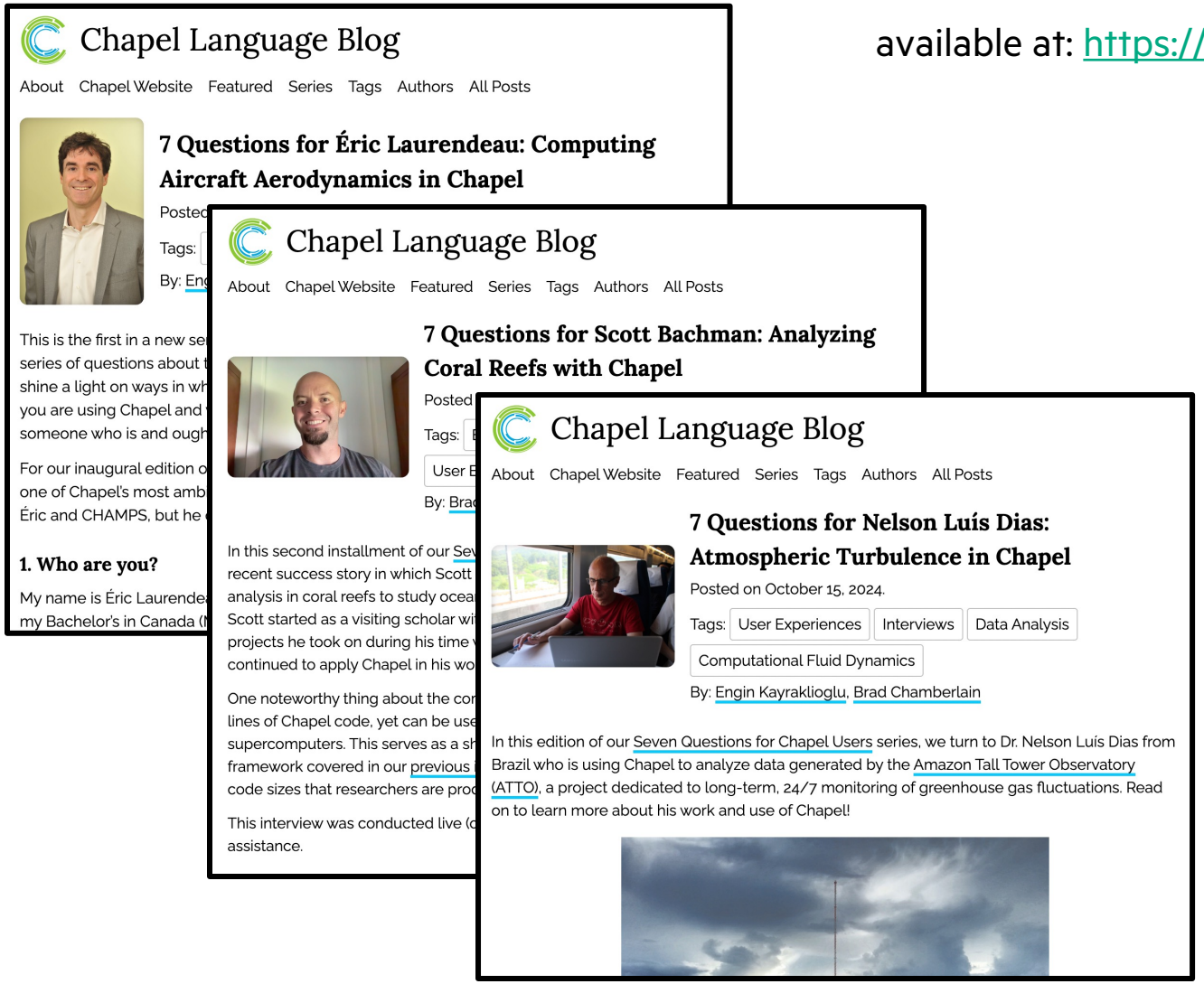
Computation: aircraft simulation
Code size: 100,000+ lines
Systems: desktops, HPC systems

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)

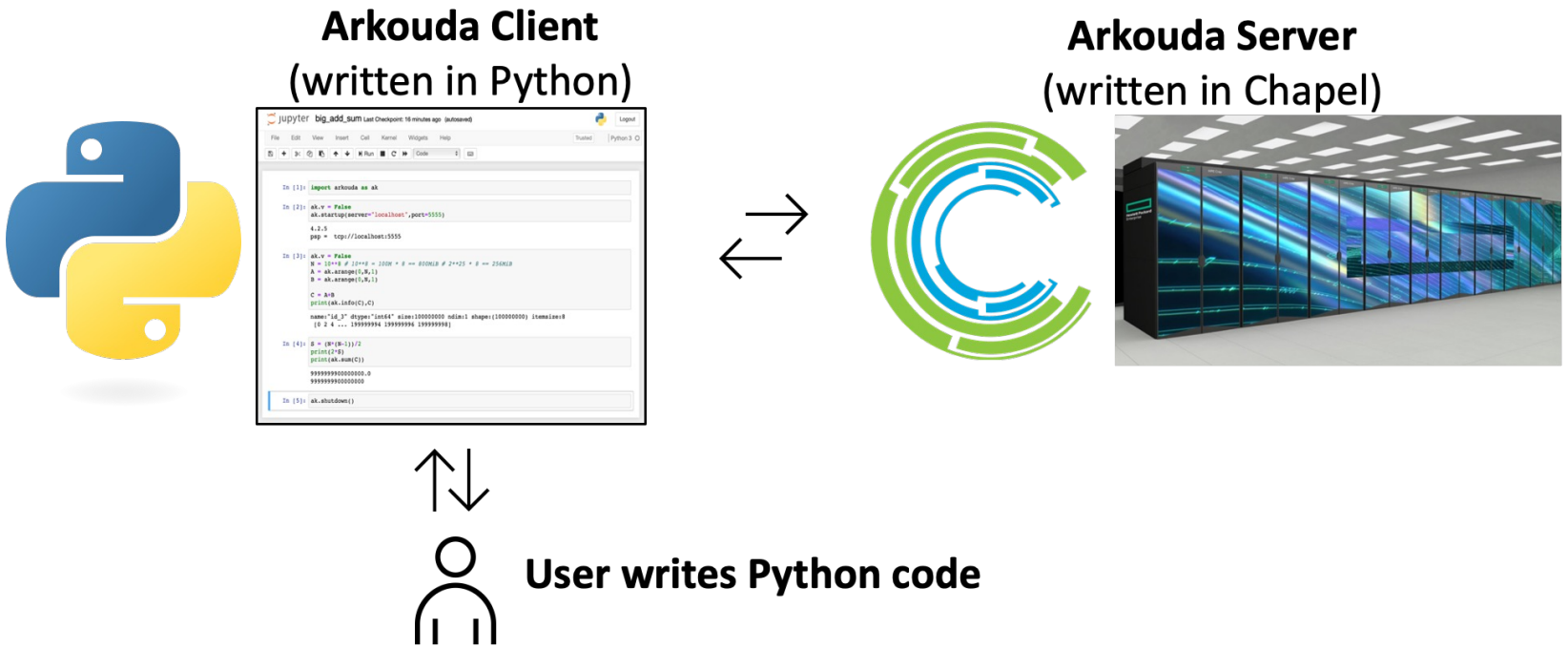
Chapel Blog Series: 7 Questions for Chapel Users



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

What is Arkouda?

Q: “What is Arkouda?”



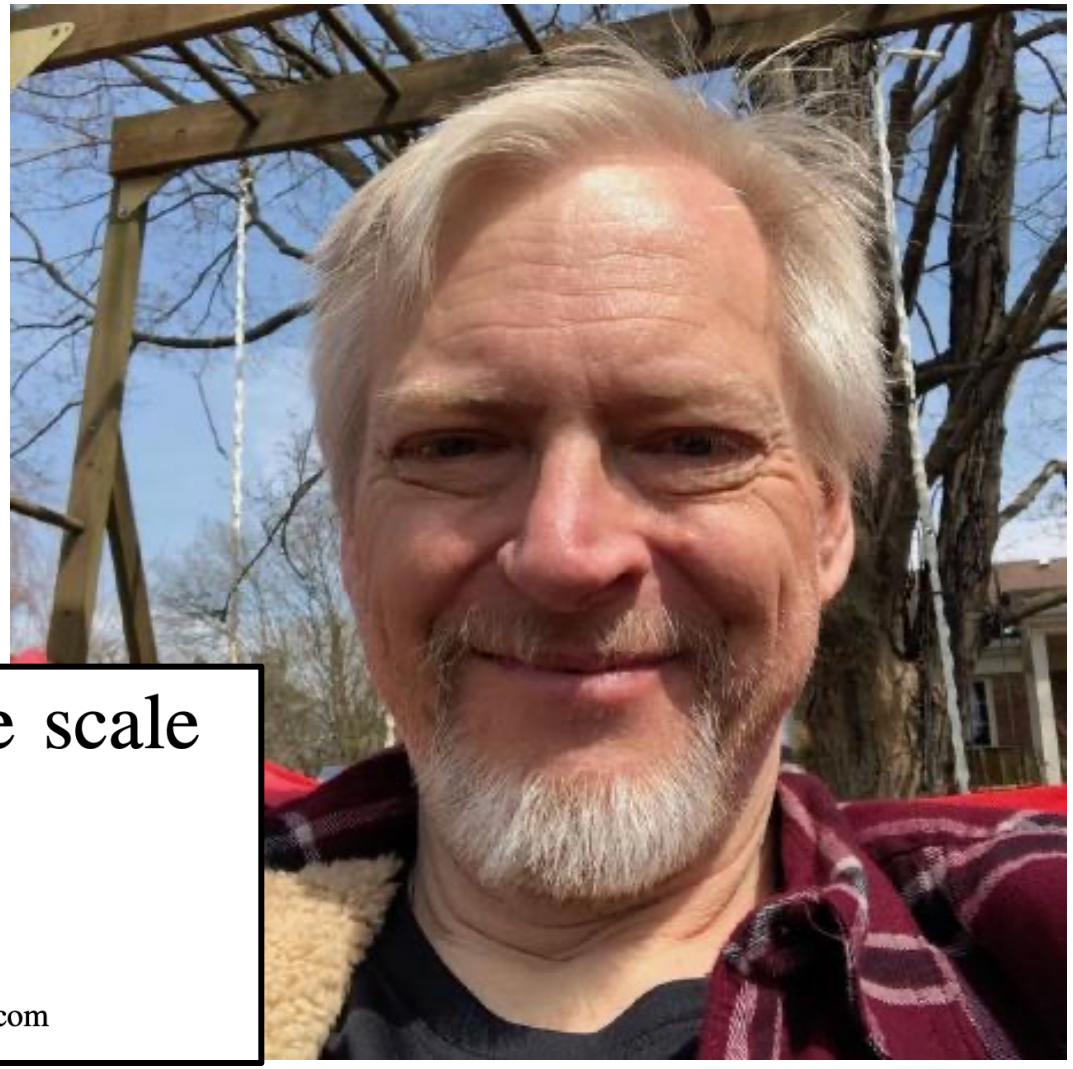
A1: “A scalable version of NumPy / Pandas for data scientists”

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: *mhmerrill@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)

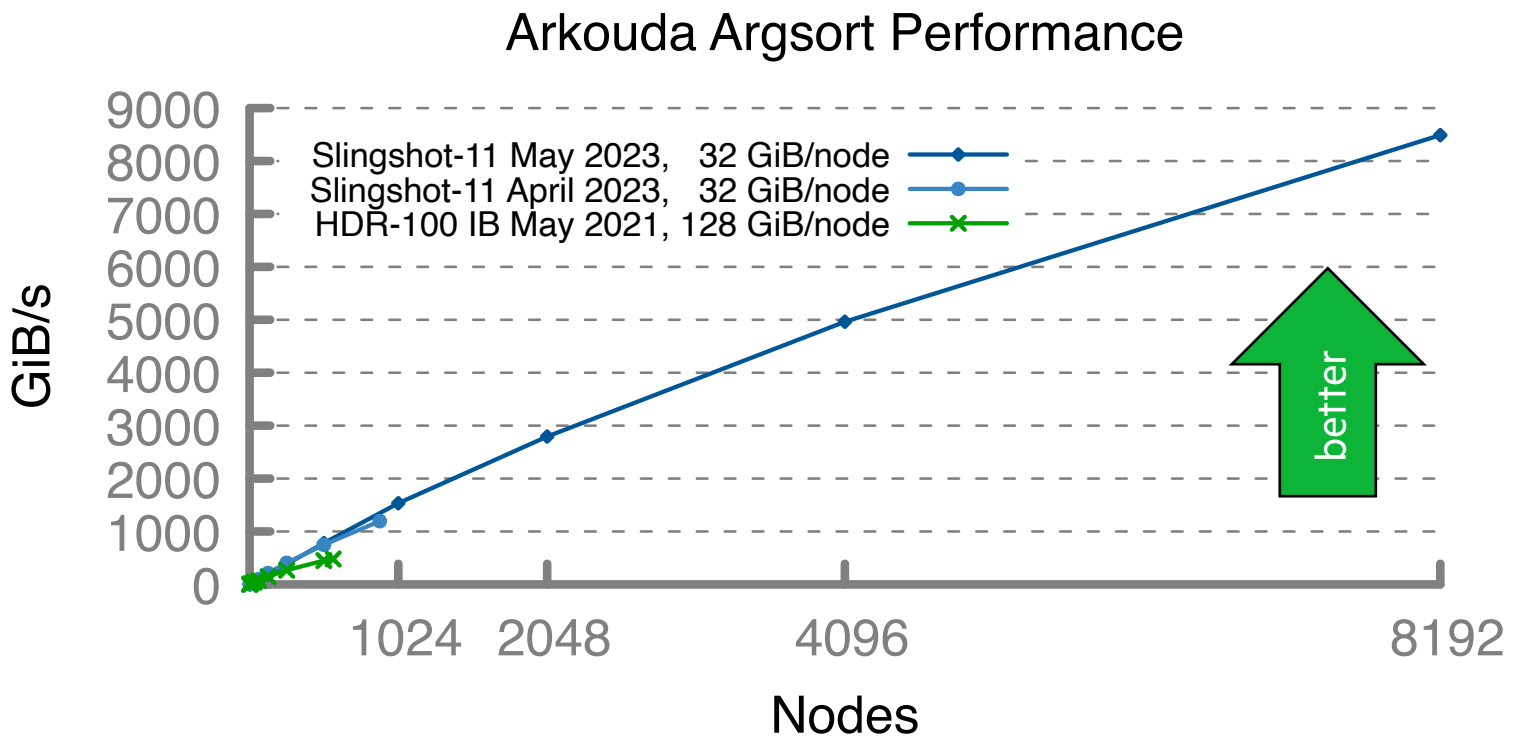
- Slingshot-11 network (200 Gb/s)
- 8192 compute nodes
- 256 TiB of 8-byte values
- ~8500 GiB/s (~31 seconds)

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:

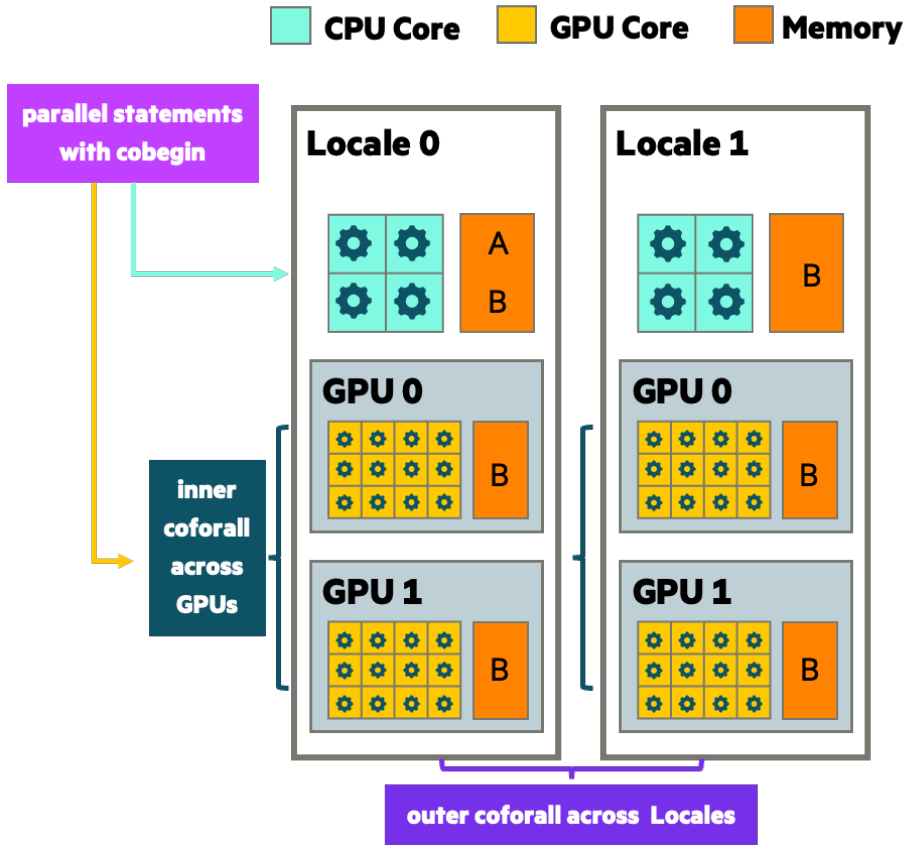
```
/* intIndex "a[int]" response to __getitem__(int) */
@arkouda.registerND(cmd_prefix="[int]")
proc intIndexMsg(cmd: string, msgArgs: borrowed MessageArgs, st: borrowed SymTab, param nd: int): MsgTuple throws {
    param pn = Reflection.getRoutineName();
    var repMsg: string; // response message
    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




```
var A: [1..n, 1..n] real;
coforall l in Locales do on l {
 cobegin {
  coforall g in here.gpus do on g {
    var B: [1..n, 1..n] real;
    B = 2;
    A = B;
  }
  {
    var B: [1..n, 1..n] real;
    B = 2;
    A = B;
  }
 }
}
writeln(A);
```

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

VIRGINIA TECH

SyNeRGy
synergy.cs.vt.edu

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

Act I: Parallelism is everywhere ...

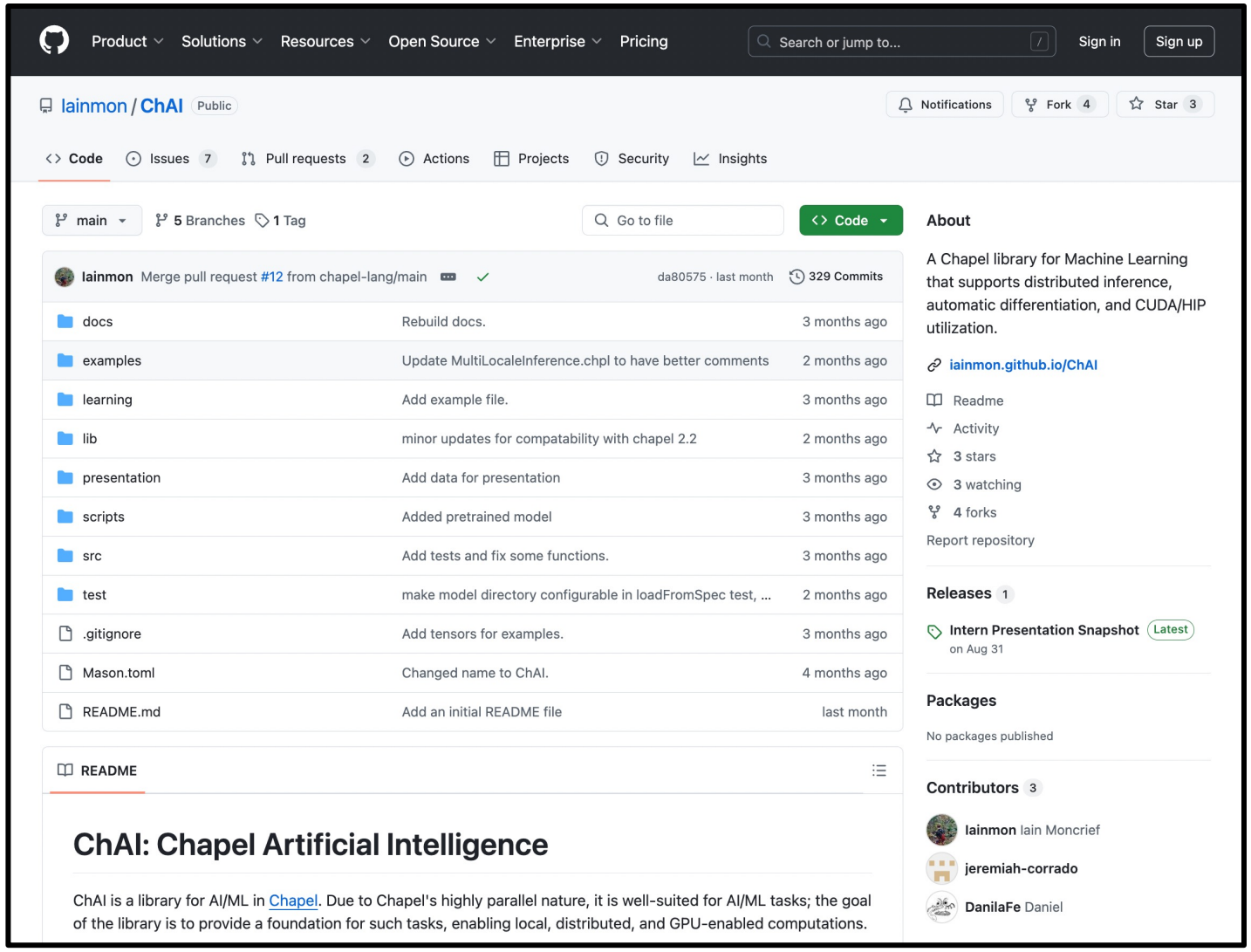
Post-serial or "serial with sprinkles"

- 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**)

ChAI: Chapel AI

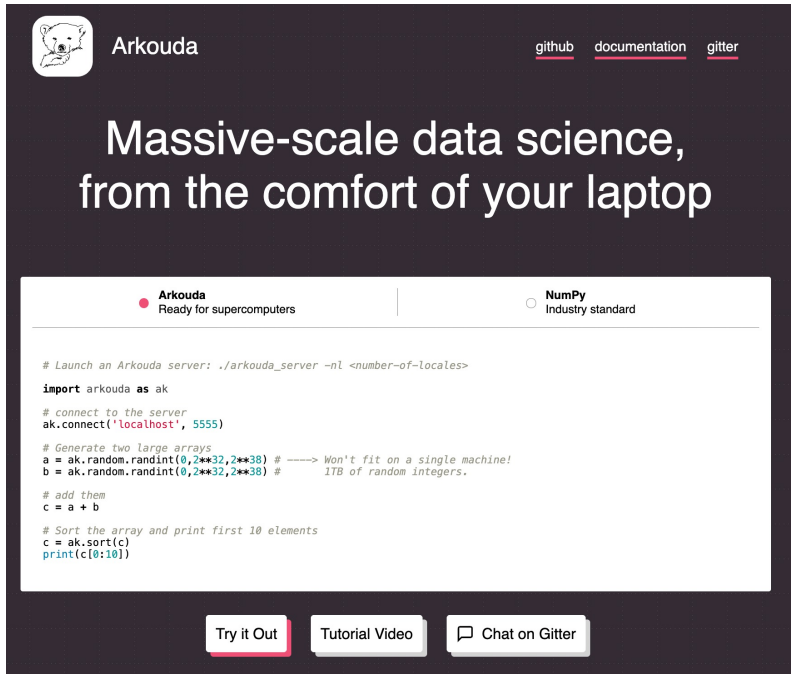
- A native PyTorch-esque Chapel module
  - Supports tensors, training, and inference
  - Runs locally, distributed, and/or on GPUs
  - Can be integrated into other HPC codes – e.g., Arkouda

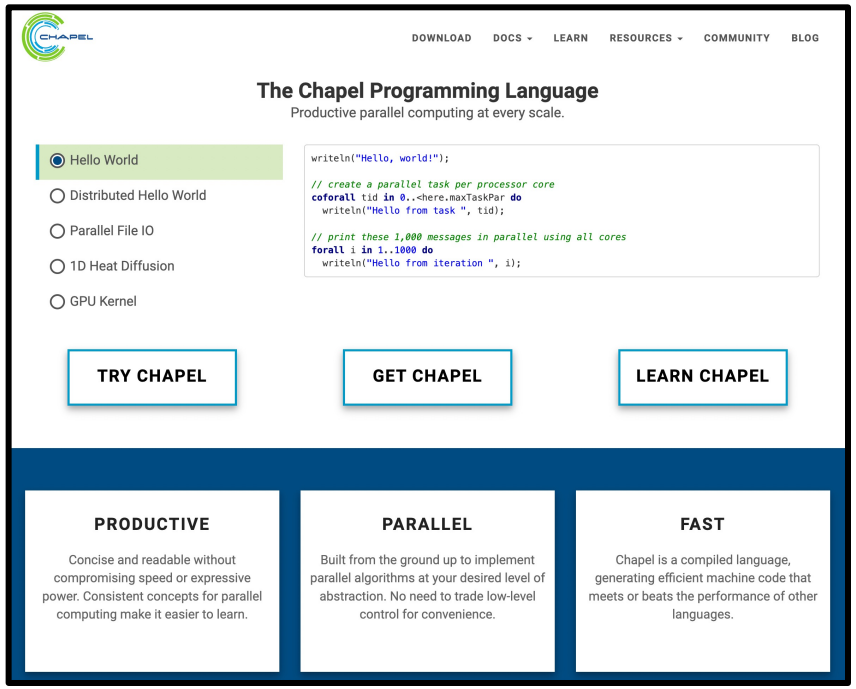
- Implemented by Iain Moncrief
  - junior at Oregon State University
  - notably, written in one summer internship



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
- **…**





Thank you

https://chapel-lang.org
@ChapelLanguage