Interactive, HPC-scale Exploratory Data Analysis in Arkouda: Past Successes and Future Challenges

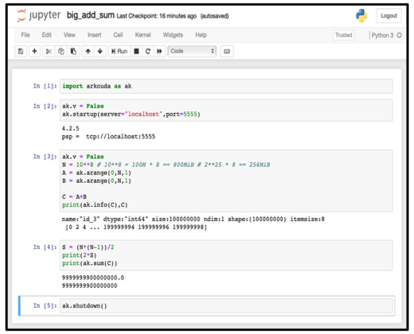
Brad Chamberlain, Advanced Programming Team, HPE

Productive, Performant Software for Large-Scale Scientific Data Analysis, SLAC
October 21, 2025

What is Arkouda?

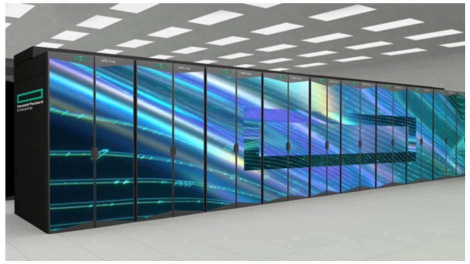
Q: “What is Arkouda?”

Arkouda Client
(written in Python)


jupyter big_add_sum Last Checkpoint: 16 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: import arkouda as ak

In [2]: ak.v = False
        ak.startup(server="localhost",port=5555)

        4.2.5
        psp =  tcp://localhost:5555

In [3]: ak.v = False
        N = 10**8 # 10**8 = 100M * 8 == 800MiB # 2**25 * 8 == 256MiB
        A = ak.arange(0,N,1)
        B = ak.arange(0,N,1)

        C = A+B
        print(ak.info(C),C)

        name:"id_3" dtype:"int64" size:100000000 ndim:1 shape:(100000000) itemsize:8
         [0 2 4 ... 199999994 199999996 199999998]

In [4]: S = (N*(N-1))/2
        print(2*S)
        print(ak.sum(C))

        9999999900000000.0
        9999999900000000

In [5]: ak.shutdown()
```

Arkouda Server
(written in Chapel)



User writes Python code
making familiar NumPy/Pandas calls

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

A2: “An extensible framework for using supercomputers interactively from Python”

Key Properties of Arkouda

- **Columnar:** represents dataframes using a distributed array per column
- **Extensible:** new features can be added to the server and/or client
  - e.g., NJIT's Arachne extension for graph analytics
- **Open-Source:** developed on GitHub, released under the MIT license
- **Portable:** runs on virtually any system (laptop, cluster, cloud instance, supercomputer)
- **Interactive:** operations are designed to complete in seconds to small numbers of minutes
- **Scalable:** has scaled to hundreds of TB, thousands of computes nodes, and over a million processor cores

Performance and Productivity: Arkouda Argsort

HPE Cray EX ◆—◆

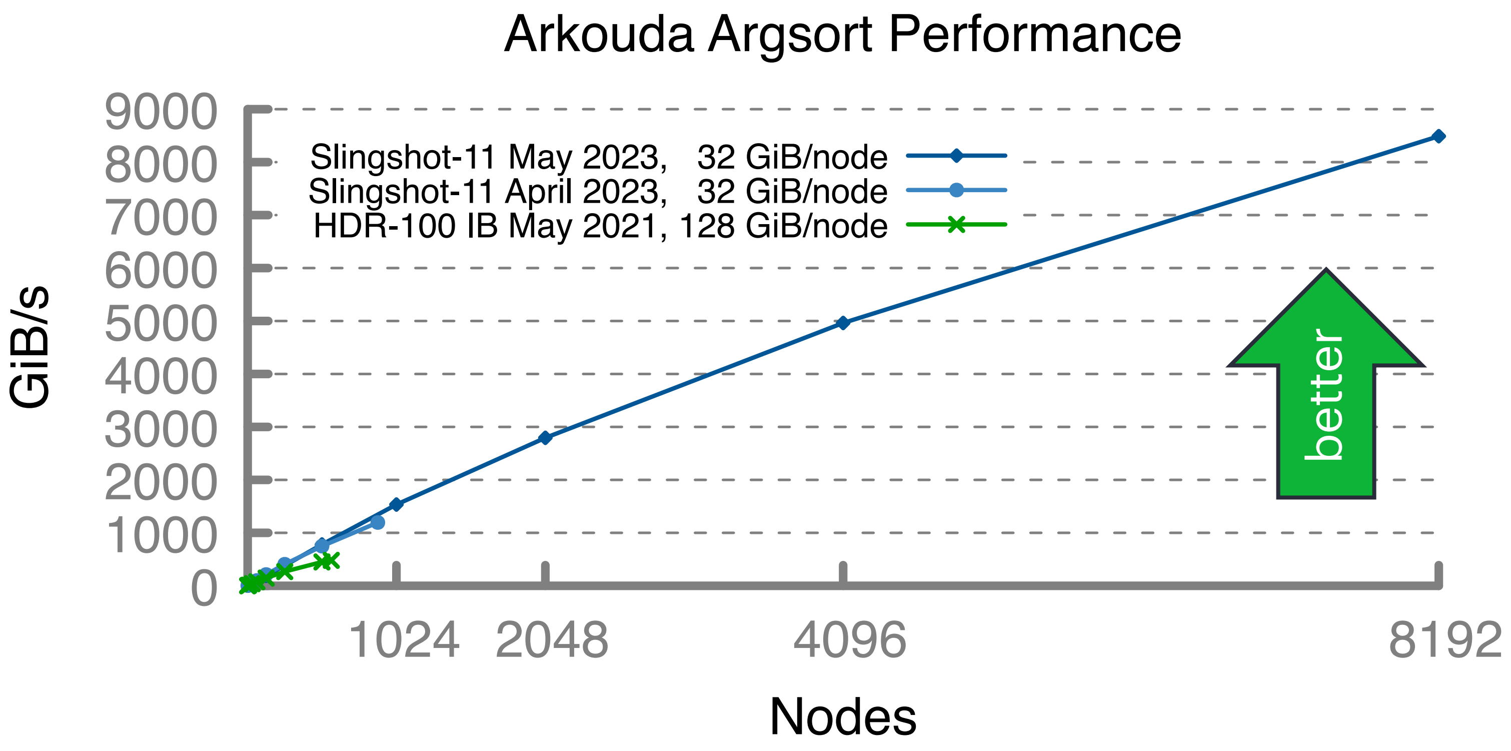
- Slingshot-11 network (200 Gb/s)
- 8192 compute nodes
- 256 TiB of 8-byte values
- ~8500 GiB/s (~31 seconds)

HPE Cray EX ●—●

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

HPE Apollo ✕—✕

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



Implemented using ~100 lines of Chapel

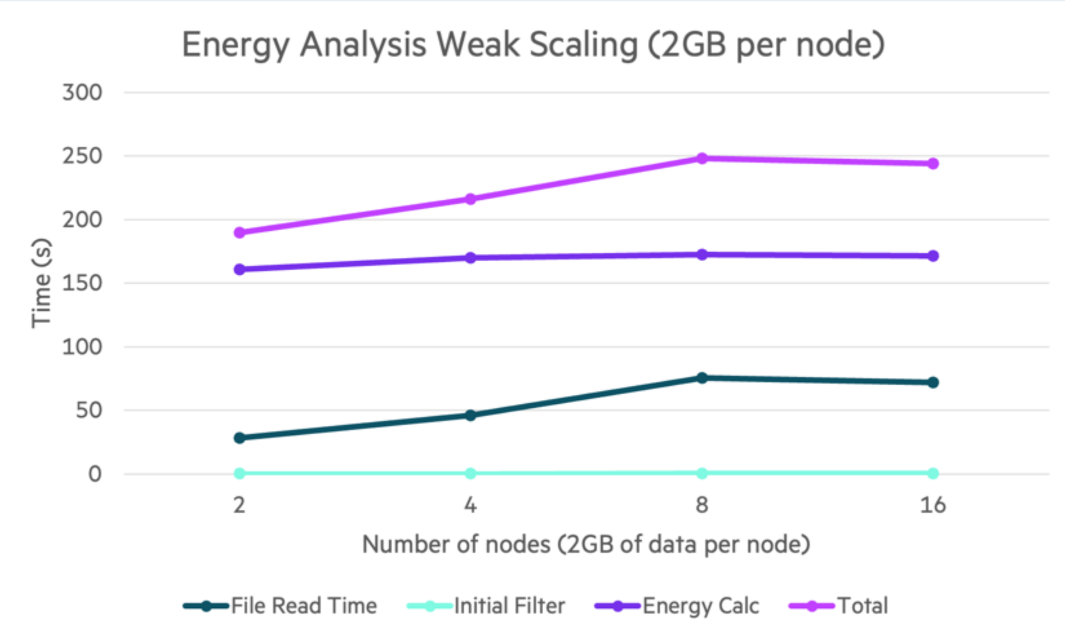
Arkouda/Pandas Comparison

— A collaboration with ORNL to analyze server telemetry data

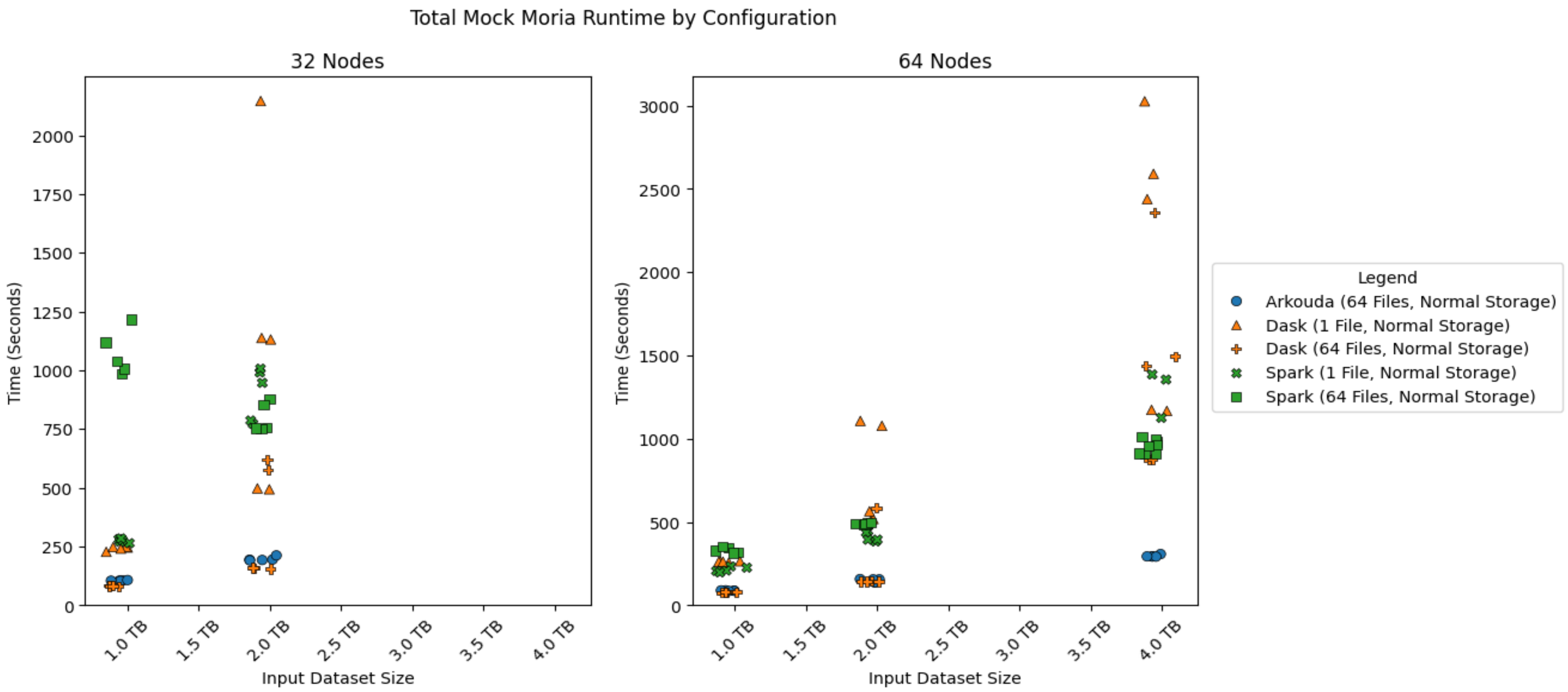
- Goal: to understand the impact of energy capping on application performance

— Translated ORNL Pandas script into Arkouda

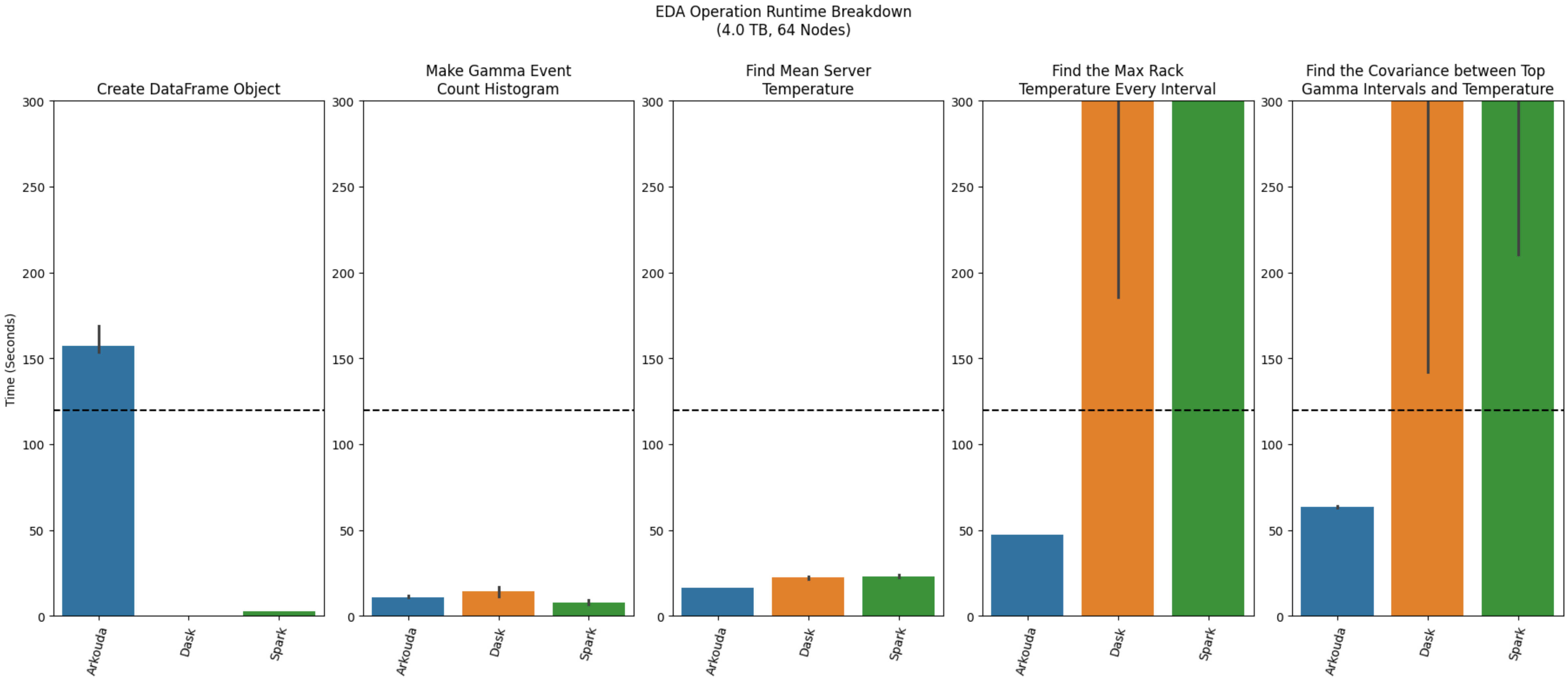
- Using the same data on a single node, Arkouda **outperformed Pandas by ~3.5x**
- Moreover, the same script shows **promising weak scaling** enabling **much larger data** to be analyzed



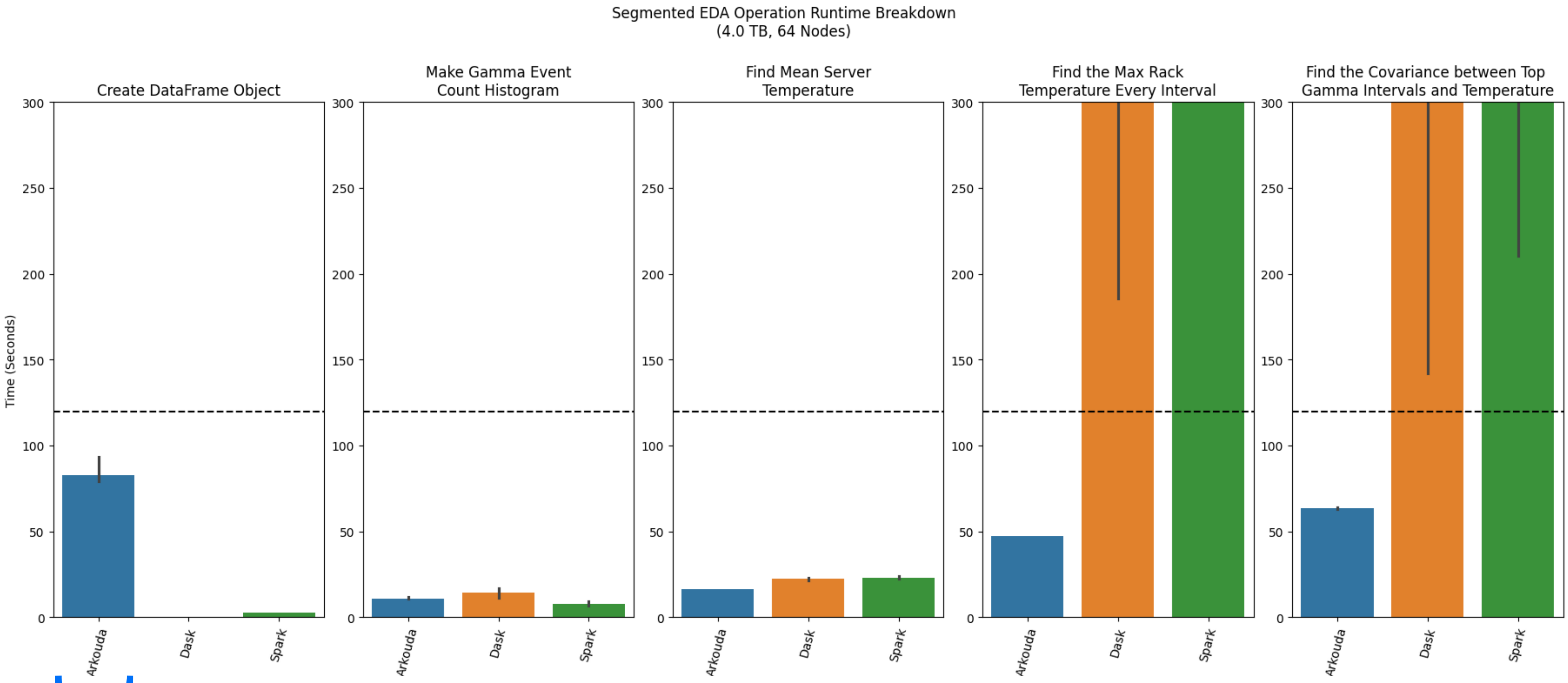
Arkouda/Dask/Spark Comparison



Arkouda/Dask/Spark Comparison: 64 nodes w/ 4 TB

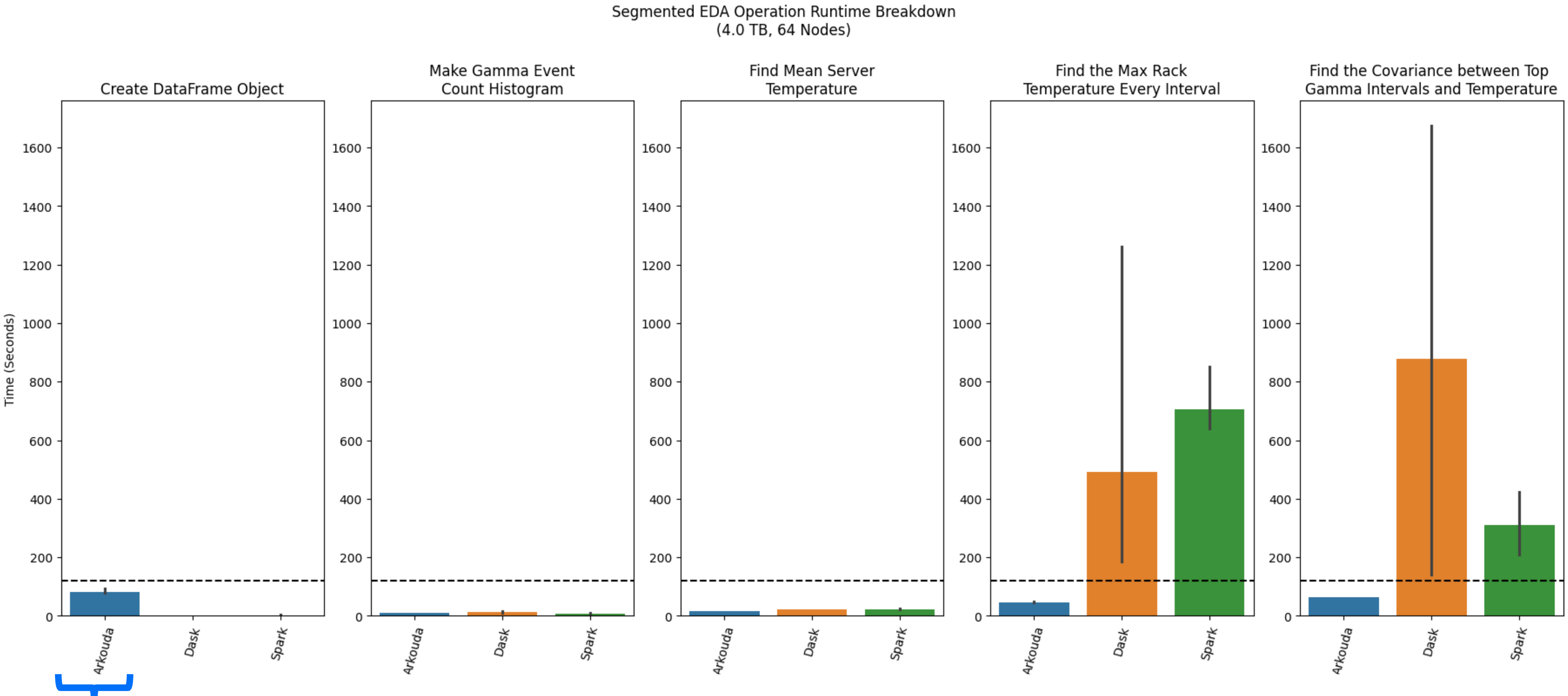


Arkouda/Dask/Spark Comparison: w/ Parquet Improvements



This specific bar has been updated to reflect recent improvements to Arkouda's Parquet IO

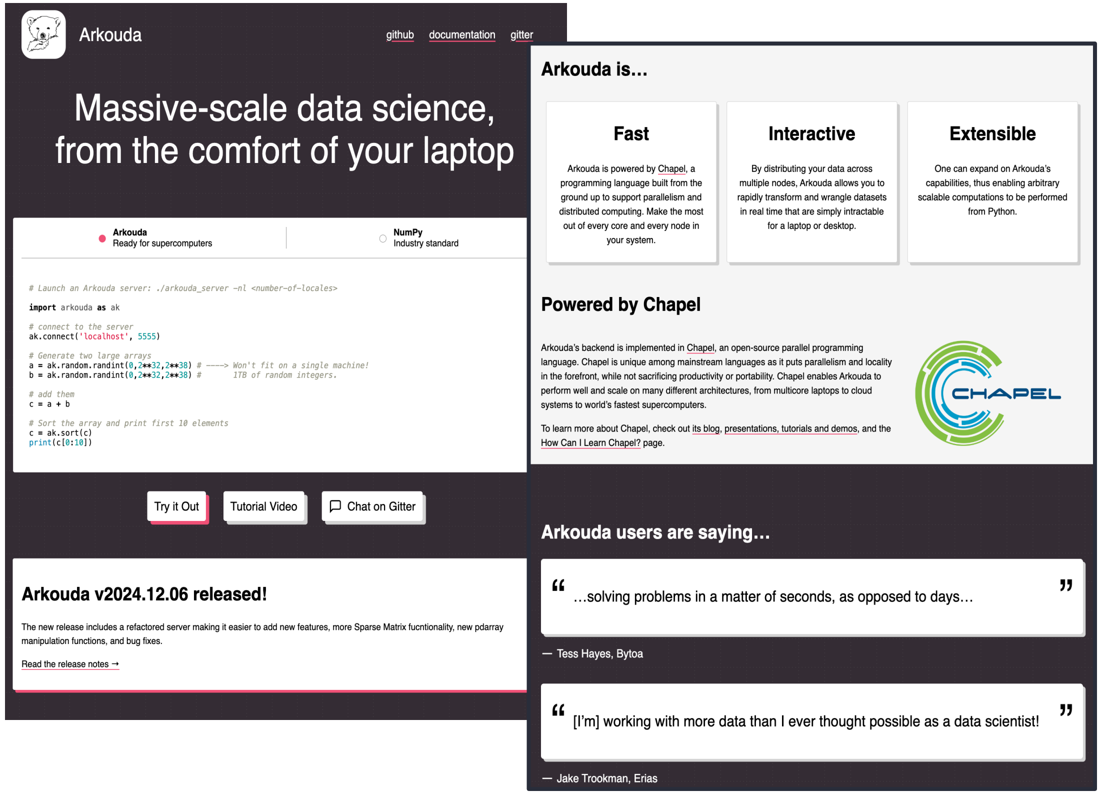
Arkouda/Dask/Spark Comparison: Zoomed out



This specific bar has been updated to reflect recent improvements to Arkouda’s Parquet IO

For More Information on Arkouda

Arkouda website:



Massive-scale data science, from the comfort of your laptop

```
# Launch an Arkouda server: ./arkouda_server -nl <number-of-locales>

import arkouda as ak

# connect to the server
ak.connect('localhost', 5555)

# Generate two large arrays
a = ak.random.randint(0,2**32,2**38) # ----> Won't fit on a single machine!
b = ak.random.randint(0,2**32,2**38) #       1TB of random integers.

# add them
c = a + b

# Sort the array and print first 10 elements
c = ak.sort(c)
print(c[0:10])
```

Arkouda v2024.12.06 released!

The new release includes a refactored server making it easier to add new features, more Sparse Matrix fucntionality, new pdarray manipulation functions, and bug fixes.

Read the release notes →

Arkouda is...

Fast

Arkouda is powered by Chapel, a programming language built from the ground up to support parallelism and distributed computing. Make the most out of every core and every node in your system.

Interactive

By distributing your data across multiple nodes, Arkouda allows you to rapidly transform and wrangle datasets in real time that are simply intractable for a laptop or desktop.

Extensible

One can expand on Arkouda's capabilities, thus enabling arbitrary scalable computations to be performed from Python.

Powered by Chapel

Arkouda's backend is implemented in Chapel, an open-source parallel programming language. Chapel is unique among mainstream languages as it puts parallelism and locality in the forefront, while not sacrificing productivity or portability. Chapel enables Arkouda to perform well and scale on many different architectures, from multicore laptops to cloud systems to world's fastest supercomputers.

To learn more about Chapel, check out its blog, presentations, tutorials and demos, and the How Can I Learn Chapel? page.

Arkouda users are saying...

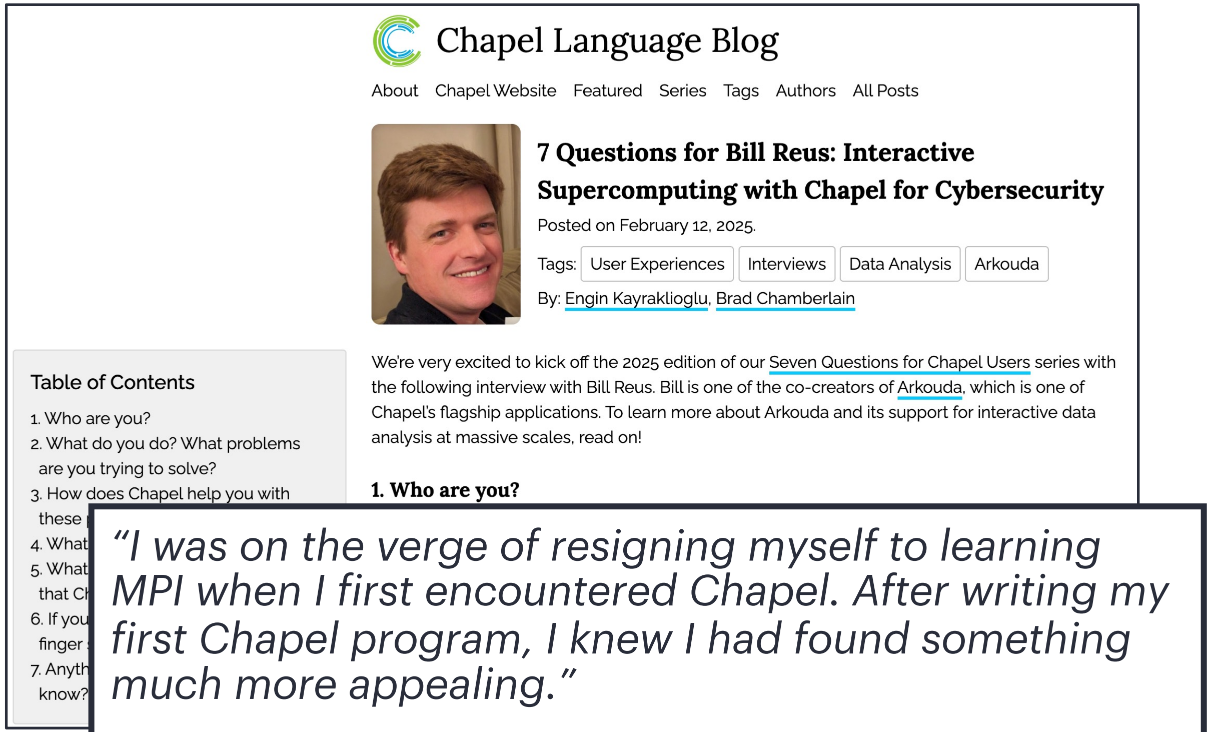
> "...solving problems in a matter of seconds, as opposed to days..."

— Tess Hayes, Bytoa

> "[I'm] working with more data than I ever thought possible as a data scientist!"

— Jake Trookman, Erias

Interview with founding co-developer, Bill Reus:



7 Questions for Bill Reus: Interactive Supercomputing with Chapel for Cybersecurity

Posted on February 12, 2025.

Tags: User Experiences | Interviews | Data Analysis | Arkouda

By: Engin Kayraklioglu, Brad Chamberlain

Table of Contents

We're very excited to kick off the 2025 edition of our Seven Questions for Chapel Users series with the following interview with Bill Reus. Bill is one of the co-creators of Arkouda, which is one of Chapel's flagship applications. To learn more about Arkouda and its support for interactive data analysis at massive scales, read on!

1. Who are you?

"I was on the verge of resigning myself to learning MPI when I first encountered Chapel. After writing my first Chapel program, I knew I had found something much more appealing."

...

"Chapel's separation of concerns immediately felt like the most natural way to think about large-scale computing. I would highly encourage anyone wanting to get into HPC programming to start with Chapel."

What is Chapel?

Chapel: A modern parallel programming language

- Portable & scalable
- Open-source & collaborative
- An HPSF / Linux Foundation project

HPSF

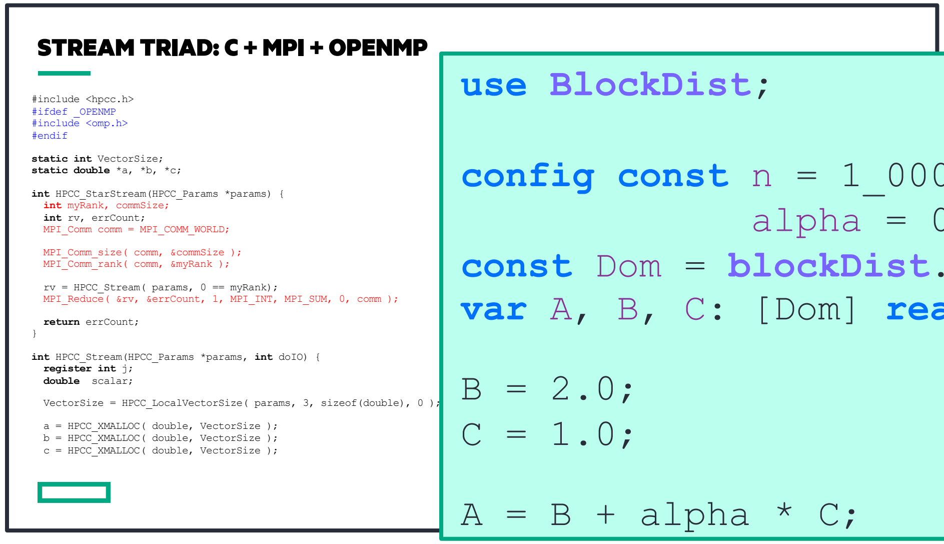
HIGH PERFORMANCE
SOFTWARE FOUNDATION

CHAPEL

Goals:

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

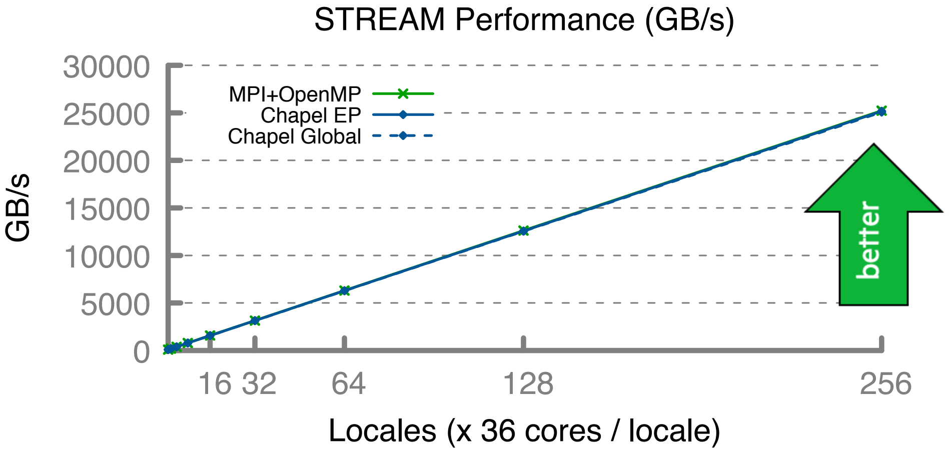
HPCC Stream Triad / RA: C+MPI+OpenMP vs. Chapel

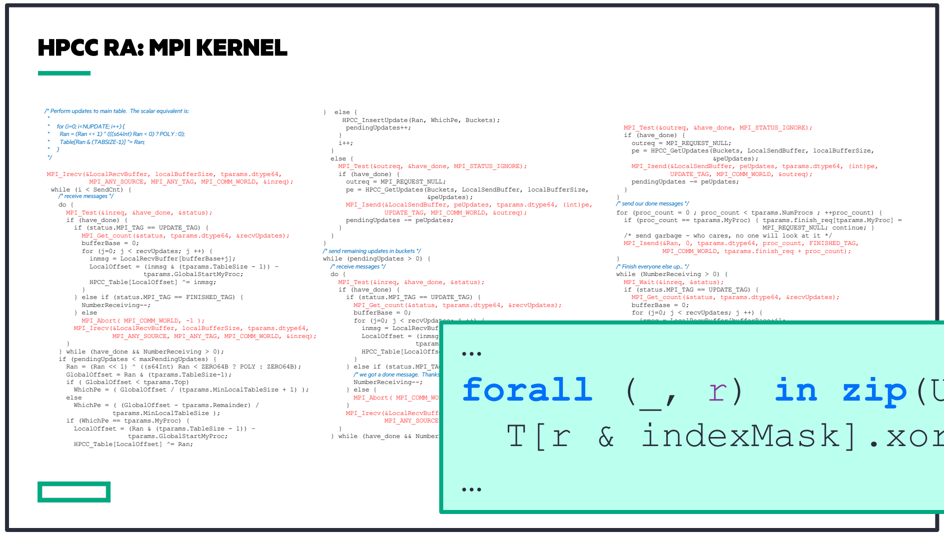


STREAM TRIAD: C + MPI + OPENMP

```
#include <hpcc.h>
#ifdef _OPENMP
#include <omp.h>
#endif

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

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

  MPI_Comm_size( comm, &commSize );
  MPI_Comm_rank( comm, &myRank );

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

  return errCount;
}

int HPCC_Stream(HPCC_Params *params, int doIO) {
  register int j;
  double  scalar;

  VectorSize = HPCC_LocalVectorSize( params, 3, sizeof(double), 0 );

  a = HPCC_XMALLOC( double, VectorSize );
  b = HPCC_XMALLOC( double, VectorSize );
  c = HPCC_XMALLOC( double, VectorSize );
```

```
use BlockDist;

config const n = 1_000_000,
             alpha = 0.01;
const Dom = blockDist.createDomain({1..n});
var A, B, C: [Dom] real;

B = 2.0;
C = 1.0;

A = B + alpha * C;
```





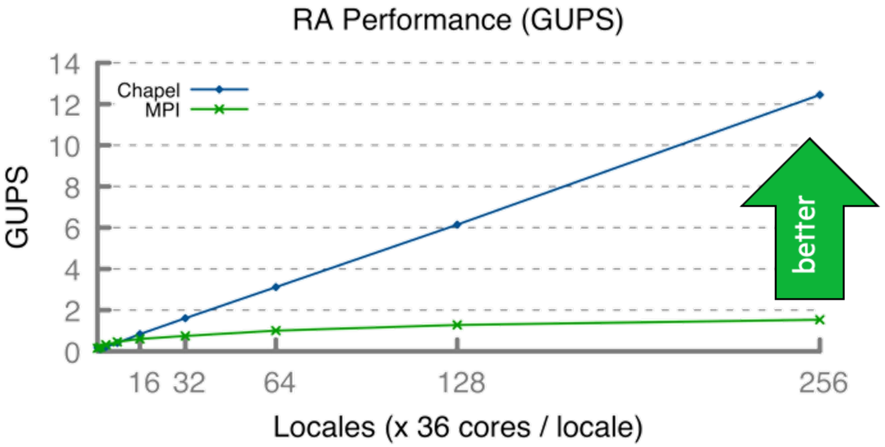
HPCC RA: MPI KERNEL

```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *    for (i=0; i<NUPDATE; i++) {
 *      Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *      Table[Ran & (TABSIZE-1)] ^= Ran;
 *    }
 */

MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
          MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
 while (i < SendCnt) {
   /* receive messages */
   do {
     MPI_Test(&inreq, &have_done, &status);
     if (have_done) {
       if (status.MPI_TAG == UPDATE_TAG) {
         MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
         bufferBase = 0;
         for (j=0; j < recvUpdates; j ++) {
           inmsg = LocalRecvBuffer[bufferBase+j];
           LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                         tparams.GlobalStartMyProc;
           HPCC_Table[LocalOffset] ^= inmsg;
         }
       } else if (status.MPI_TAG == FINISHED_TAG) {
         NumberReceiving--;
       } else
         MPI_Abort( MPI_COMM_WORLD, -1 );
       MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                 MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
     }
   } while (have_done && NumberReceiving > 0);
   if (pendingUpdates < maxPendingUpdates) {
     Ran = (Ran << 1) ^ ((s64Int) Ran < ZERO64B ? POLY : ZERO64B);
     GlobalOffset = Ran & (tparams.TableSize-1);
     if ( GlobalOffset < tparams.Top)
       WhichPe = ( GlobalOffset / (tparams.MinLocalTableSize + 1) );
     else
       WhichPe = ( (GlobalOffset - tparams.Remainder) /
                   tparams.MinLocalTableSize );
     if (WhichPe == tparams.MyProc) {
       LocalOffset = (Ran & (tparams.TableSize - 1)) -
                     tparams.GlobalStartMyProc;
       HPCC_Table[LocalOffset] ^= Ran;
```

```
    }  else {
      HPCC_InsertUpdate(Ran, WhichPe, Buckets);
      pendingUpdates++;
    }
    i++;
  }
  else {
    MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
    if (have_done) {
      outreq = MPI_REQUEST_NULL;
      pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                           &peUpdates);
      MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
                UPDATE_TAG, MPI_COMM_WORLD, &outreq);
      pendingUpdates -= peUpdates;
    }
  }
}
/* send remaining updates in buckets */
while (pendingUpdates > 0) {
  /* receive messages */
  do {
    MPI_Test(&inreq, &have_done, &status);
    if (have_done) {
      if (status.MPI_TAG == UPDATE_TAG) {
        MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
        bufferBase = 0;
        for (j=0; j < recvUpda
          inmsg = LocalRecvBuf
          LocalOffset = (inmsg
                        tparam
          HPCC_Table[LocalOffs
        }
      } else if (status.MPI_TA
        /* we got a done message.  Thanks
        NumberReceiving--;
      } else {
        MPI_Abort( MPI_COMM_WO
      }
      MPI_Irecv(&LocalRecvBuff
                MPI_ANY_SOURCE
    }
  } while (have_done && Number
```

```
  MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
  if (have_done) {
    outreq = MPI_REQUEST_NULL;
    pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                         &peUpdates);
    MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
              UPDATE_TAG, MPI_COMM_WORLD, &outreq);
    pendingUpdates -= peUpdates;
  }
}
/* send our done messages */
for (proc_count = 0 ; proc_count < tparams.NumProcs ; ++proc_count) {
  if (proc_count == tparams.MyProc) { tparams.finish_req[tparams.MyProc] =
                                      MPI_REQUEST_NULL; continue; }
  /* send garbage - who cares, no one will look at it */
  MPI_Isend(&Ran, 0, tparams.dtype64, proc_count, FINISHED_TAG,
            MPI_COMM_WORLD, tparams.finish_req + proc_count);
}
/* Finish everyone else up... */
while (NumberReceiving > 0) {
  MPI_Wait(&inreq, &status);
  if (status.MPI_TAG == UPDATE_TAG) {
    MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
    bufferBase = 0;
    for (j=0; j < recvUpdates; j ++) {
```

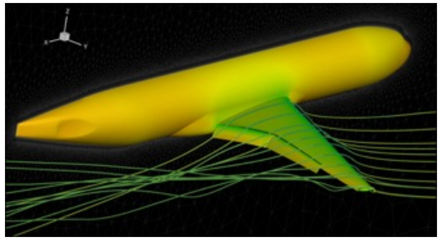
```
…
forall (_, r) in zip(Updates, RandVals()) do
  T[r & indexMask].xor(r);
…
```



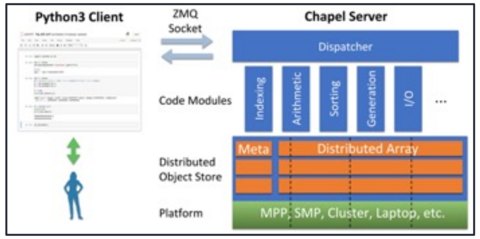
Why was Arkouda written in Chapel?

- **productivity**, readability, writability
  - Pythonic syntax is attractive to Python users who want to add features
- **parallelism** and **distributed arrays** as first-class features
- **performance:** competitive with conventional approaches
- **portability**: developed on laptop, deployed on supercomputer
- **interoperability:** can call to existing libraries

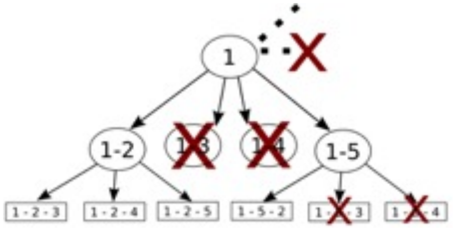
Applications of Chapel



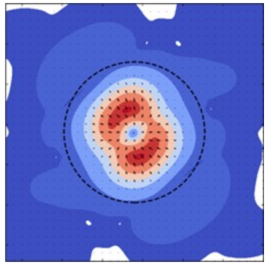
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



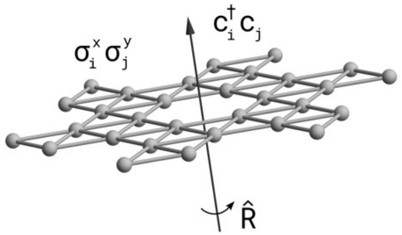
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



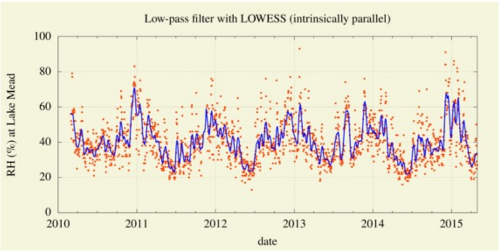
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

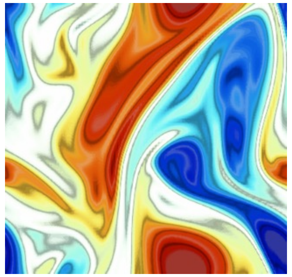


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

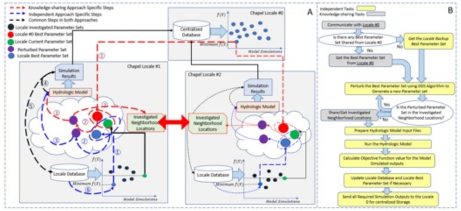


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

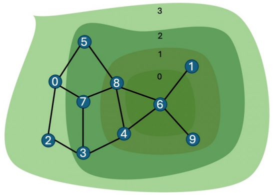
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



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

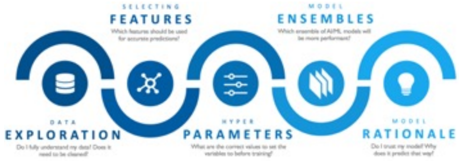


Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



Arachne Graph Analytics
Bader, Du, Rodriguez, et al.
New Jersey Institute of Technology

Modeling Ocean Carbon Dioxide Removal
Scott Bachman Brandon Neth, et al.
[C]Worthy

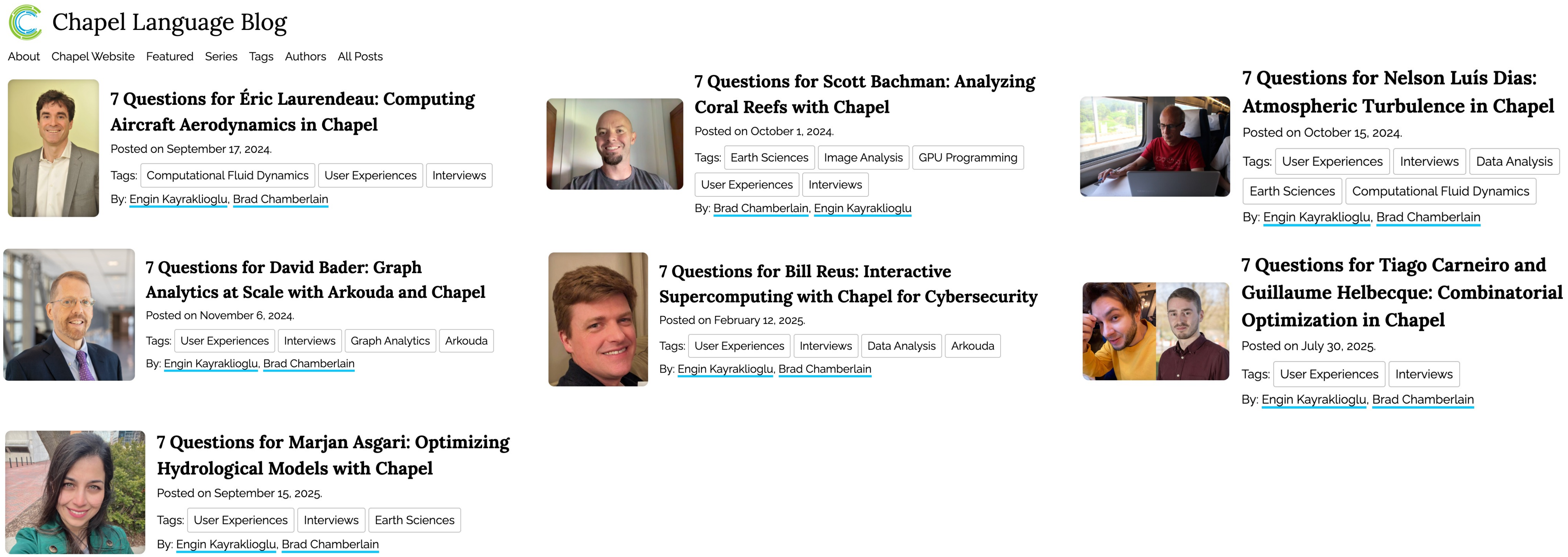


CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE

[images provided by their respective teams and used with permission]

"7 Questions with Chapel Users" Interviews

Read about users' Chapel experiences in the "*7 Questions with Chapel Users*" series on our blog



Chapel Language Blog

About Chapel Website Featured Series Tags Authors All Posts

7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel

Posted on September 17, 2024.

Tags: Computational Fluid Dynamics | User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel

Posted on October 1, 2024.

Tags: Earth Sciences | Image Analysis | GPU Programming | User Experiences | Interviews

By: Brad Chamberlain, Engin Kayraklioglu

7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel

Posted on October 15, 2024.

Tags: User Experiences | Interviews | Data Analysis | Earth Sciences | Computational Fluid Dynamics

By: Engin Kayraklioglu, Brad Chamberlain

7 Questions for David Bader: Graph Analytics at Scale with Arkouda and Chapel

Posted on November 6, 2024.

Tags: User Experiences | Interviews | Graph Analytics | Arkouda

By: Engin Kayraklioglu, Brad Chamberlain

7 Questions for Bill Reus: Interactive Supercomputing with Chapel for Cybersecurity

Posted on February 12, 2025.

Tags: User Experiences | Interviews | Data Analysis | Arkouda

By: Engin Kayraklioglu, Brad Chamberlain

7 Questions for Tiago Carneiro and Guillaume Helbecque: Combinatorial Optimization in Chapel

Posted on July 30, 2025.

Tags: User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

7 Questions for Marjan Asgari: Optimizing Hydrological Models with Chapel

Posted on September 15, 2025.

Tags: User Experiences | Interviews | Earth Sciences

By: Engin Kayraklioglu, Brad Chamberlain

Ways to engage with the Chapel Community

Synchronous Community Events

- Project Meetings, weekly
- Deep Dive / Demo Sessions, weekly timeslot
- ChapelCon (formerly CHIUW), annually

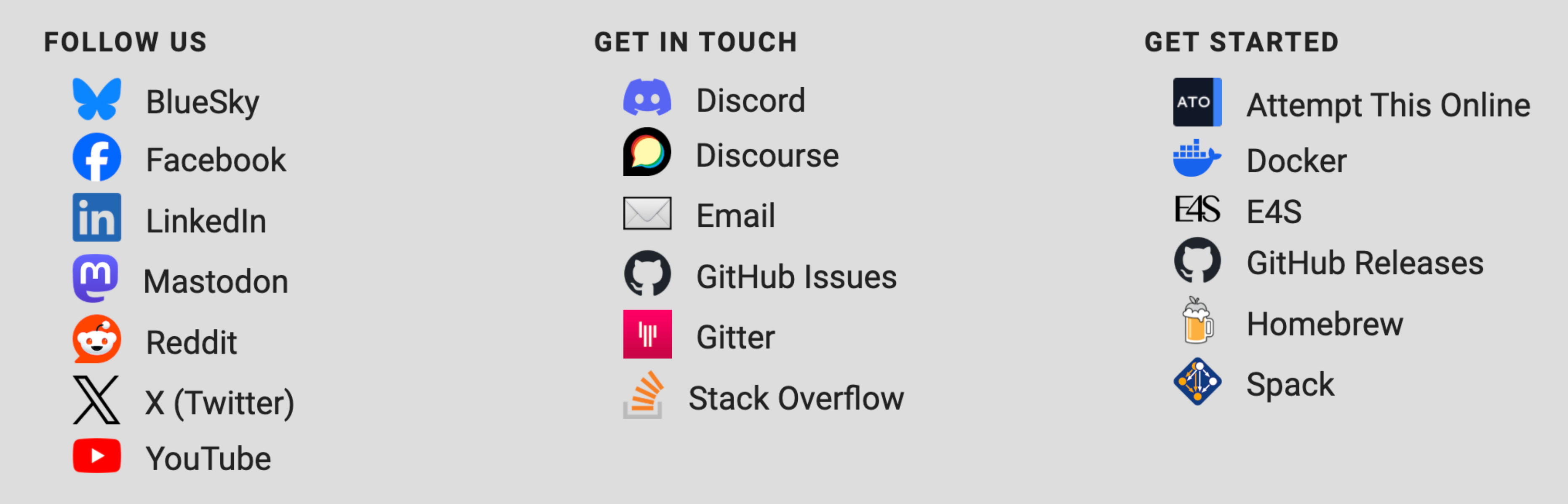
Asynchrounous Communications

- Chapel Blog, typically ~2 articles per month
- Community Newsletter, quarterly
- Announcement Emails, around big events

Social Media

Discussion Forums

Ways to Use Chapel



FOLLOW US

- BlueSky
- Facebook
- LinkedIn
- Mastodon
- Reddit
- X (Twitter)
- YouTube

GET IN TOUCH

- Discord
- Discourse
- Email
- GitHub Issues
- Gitter
- Stack Overflow

GET STARTED

- Attempt This Online
- Docker
- E4S
- GitHub Releases
- Homebrew
- Spack

(from the footer of chapel-lang.org)

Next Steps: SUF Characterizations / Speed-Dating?

Big Q: With current capabilities, can Arkouda support Scientific User Facility (SUF) workloads?

- correct file formats?
- required operations?
- performance and scalability?

If not, what is lacking, and what would be required to address them?

Next Steps: Research Questions and Challenges

GPUs:

- Would scientific data analysis (SDA) operations benefit from GPU acceleration? Or are other things a bottleneck?
- Would such use cases require new features from Arkouda/Chapel? (e.g., GPU-initiated communication?)

IO subsystems and file formats:

- What new IO systems or file formats might be beneficial, or do we have what we need?
- What changes to system-level software would be necessary to (better) leverage such IO capabilities?

Custom Hardware Accelerators:

- What role might exotic new chips play in the SDA space?
- Will these be generally programmable, or more like library operations in silicon?

Extensibility:

- How can Arkouda’s extensibility be streamlined to add new capabilities for rapidly changing requirements?
- What is required to dynamically add new Arkouda modules to a running server?

Community:

- How can HPC break its cycle of failing to broadly adopt new productive software systems?
- How should innovative HPC software be fostered and sustained over time?

Thank You

@ChapelLanguage