Interactive Data Science at Massive Scales using Python and Arkouda

Brad Chamberlain w/ Jade Abraham, Advanced Programming Team, HPE

UW Data Science Seminar
December 2, 2025

A Bit About Me

A Bit About You?

Today's talk as a research question:

"As computer scientists and HPC* experts, what can we do to enable productive data science on massive data sets?"

*HPC = High-Performance Computing

Data Science In Python at scale?

Motivation: Imagine you work with...

...Python-based data scientists

...HPC-scale data science problems to solve

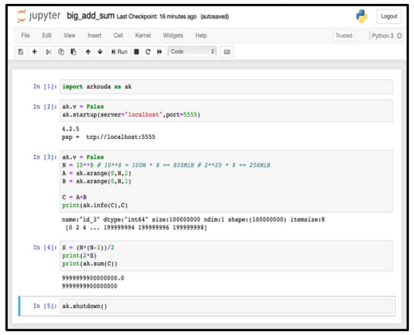
...access to HPC systems

How will you leverage your Python programmers to get your work done?

What is Arkouda?

Q: “What is Arkouda?”

Arkouda Client
(written in Python)



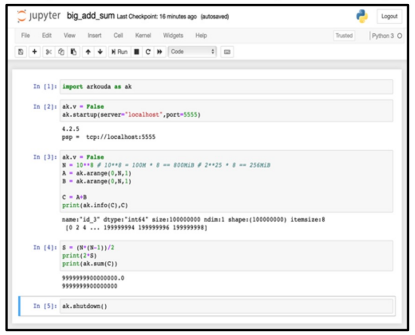
```
import arkouda as ak

def ak_argsort(N, seed):
    a = ak.randint(0, 2**64, N, dtype=ak.uint64, seed=seed)
    perm = ak.argsort(a)

    assert ak.is_sorted(a[perm])
```

User writes Python code
making familiar NumPy/Pandas calls

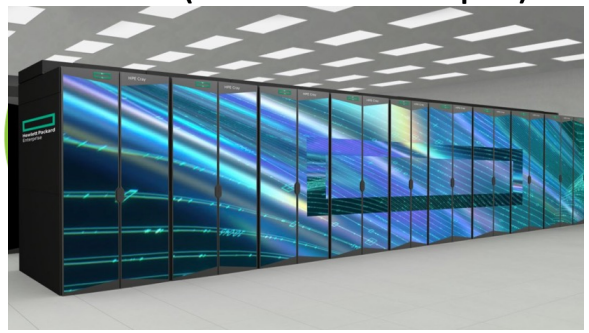
What is Arkouda?

Q: “What is Arkouda?”

Arkouda Client
(written in Python)



Arkouda Server
(written in Chapel)



User writes Python code
making familiar NumPy/Pandas calls

```
import arkouda as ak

def ak_argsort(N, seed):
    a = ak.randint(0, 2**64, N, dtype=ak.uint64, seed=seed)
    perm = ak.argsort(a)

    assert ak.is_sorted(a[perm])
```

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

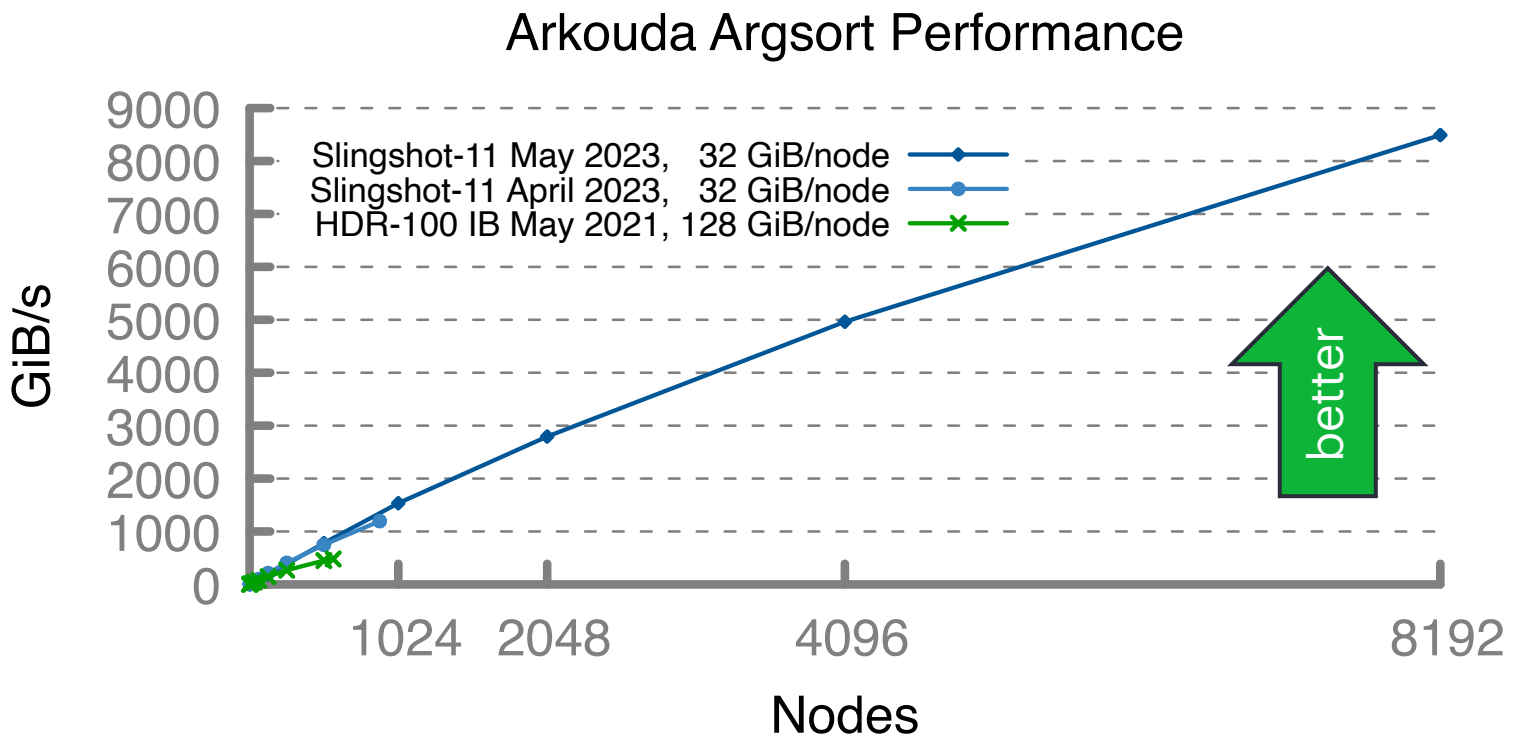
A2: “A framework for driving supercomputers interactively from Python”

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)

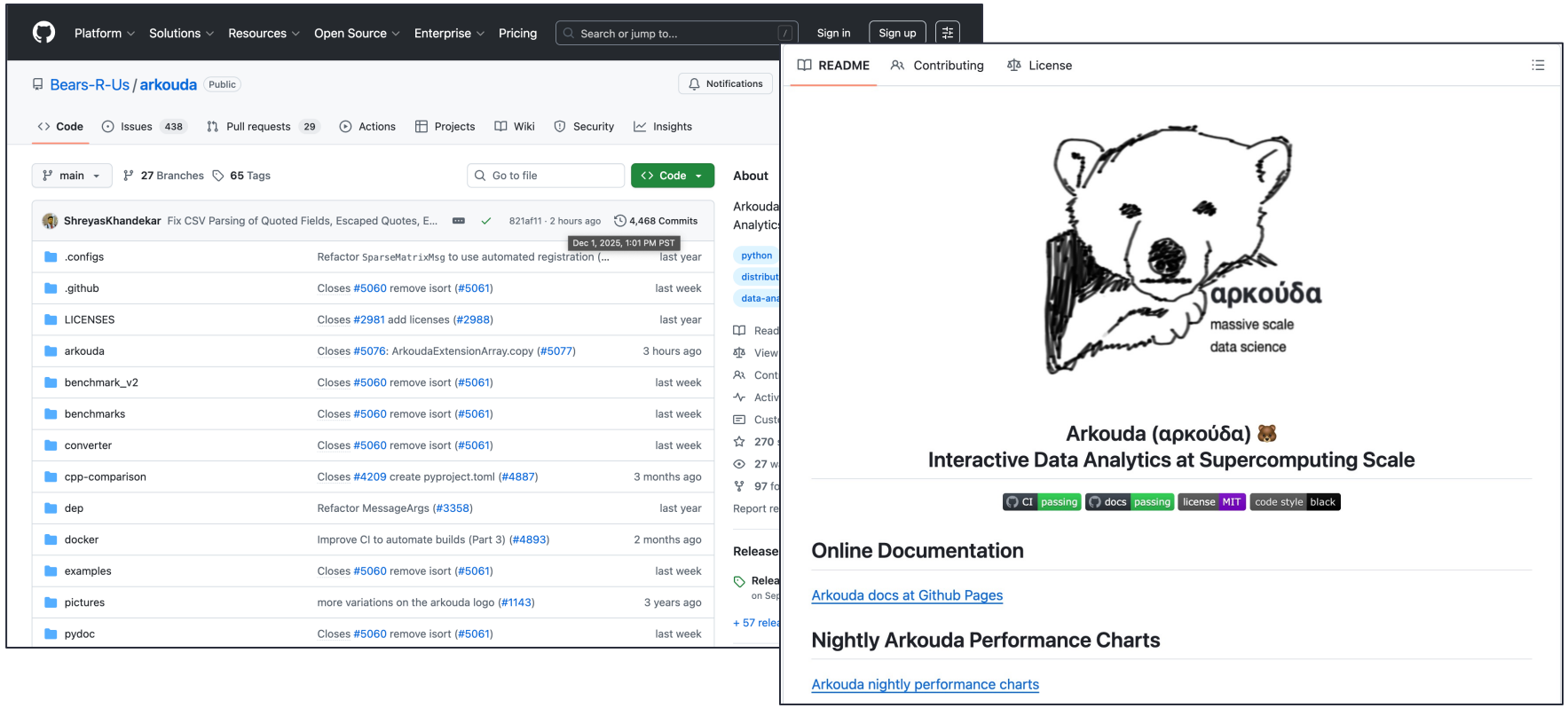
HPE Apollo
- HDR-100 InfiniBand network (100 Gb/s)
- 576 compute nodes
- 72 TiB of 8-byte values
- ~480 GiB/s (~150 seconds)



Implemented using ~100 lines of Chapel code

Key Properties of Arkouda

- **Scalable:** has scaled to hundreds of TB, thousands of computes nodes, and over a million processor cores
- **Interactive:** operations are designed to complete in seconds to small numbers of minutes
- **Portable:** runs on virtually any system (laptop, cluster, cloud instance, supercomputer)
- **Open-Source:** developed on GitHub, released under the MIT license



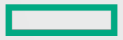
Key Properties of Arkouda

- **Scalable:** has scaled to hundreds of TB, thousands of computes nodes, and over a million processor cores
- **Interactive:** operations are designed to complete in seconds to small numbers of minutes
- **Portable:** runs on virtually any system (laptop, cluster, cloud instance, supercomputer)
- **Open-Source:** developed on GitHub, released under the MIT license
- **Columnar:** represents dataframes using a distributed array per column
  - Current I/O Formats: Parquet, CSV, HDF5
- ...

Outline

- Introduction to Arkouda
- **Performance/Scaling Comparisons**
- **Live Demo**
- **Extensibility**
- **Wrap-Up**

Arkouda Performance and Scaling Comparisons



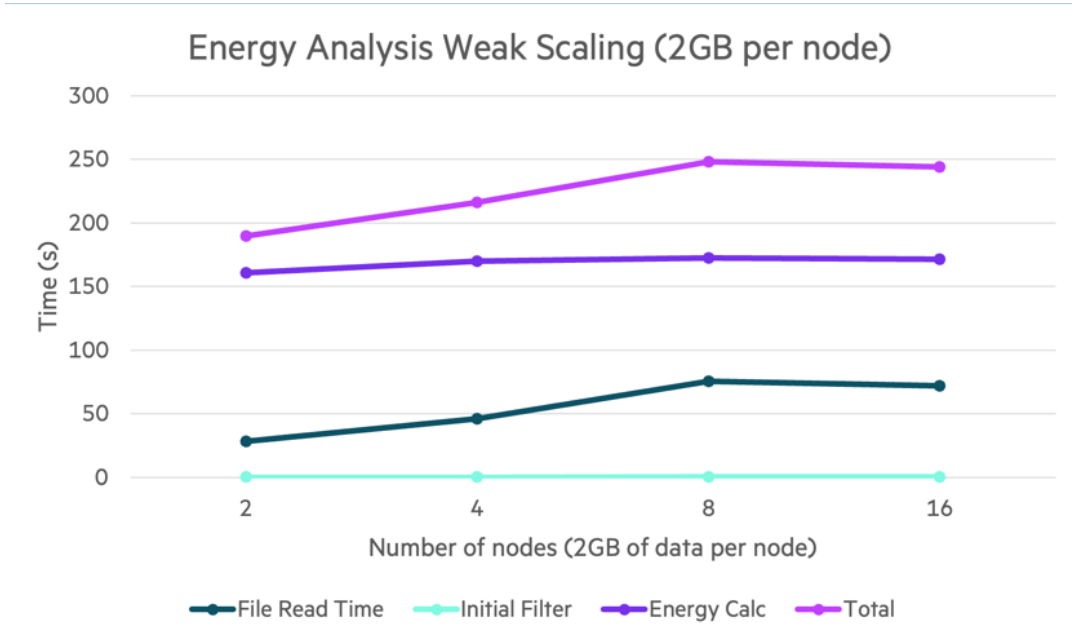
Arkouda/Pandas Comparison

Background: A collaboration with ORNL to analyze telemetry data from their Frontier supercomputer

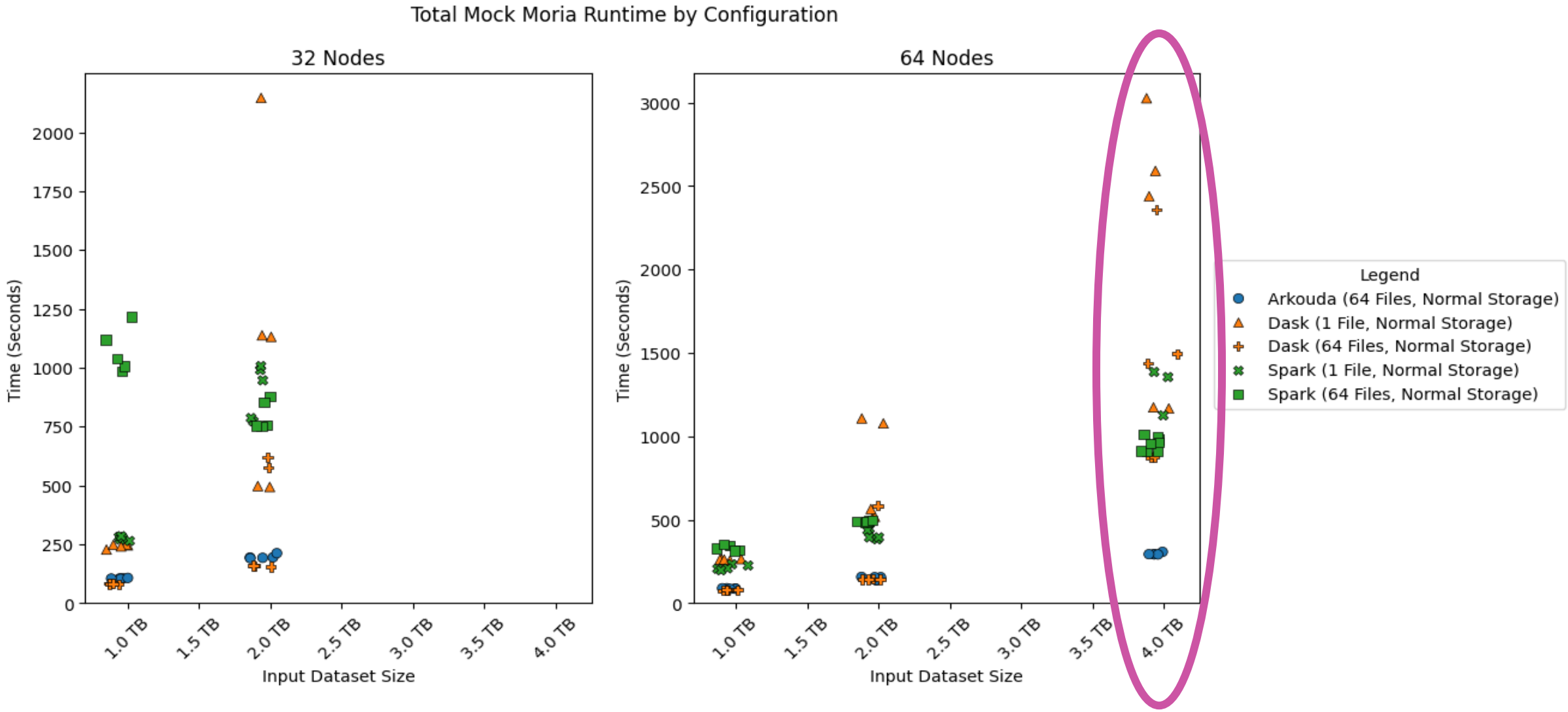
- **Goal:** to understand the application performance impact of energy-capping GPUs

Experience: Translated ORNL Pandas script into Arkouda

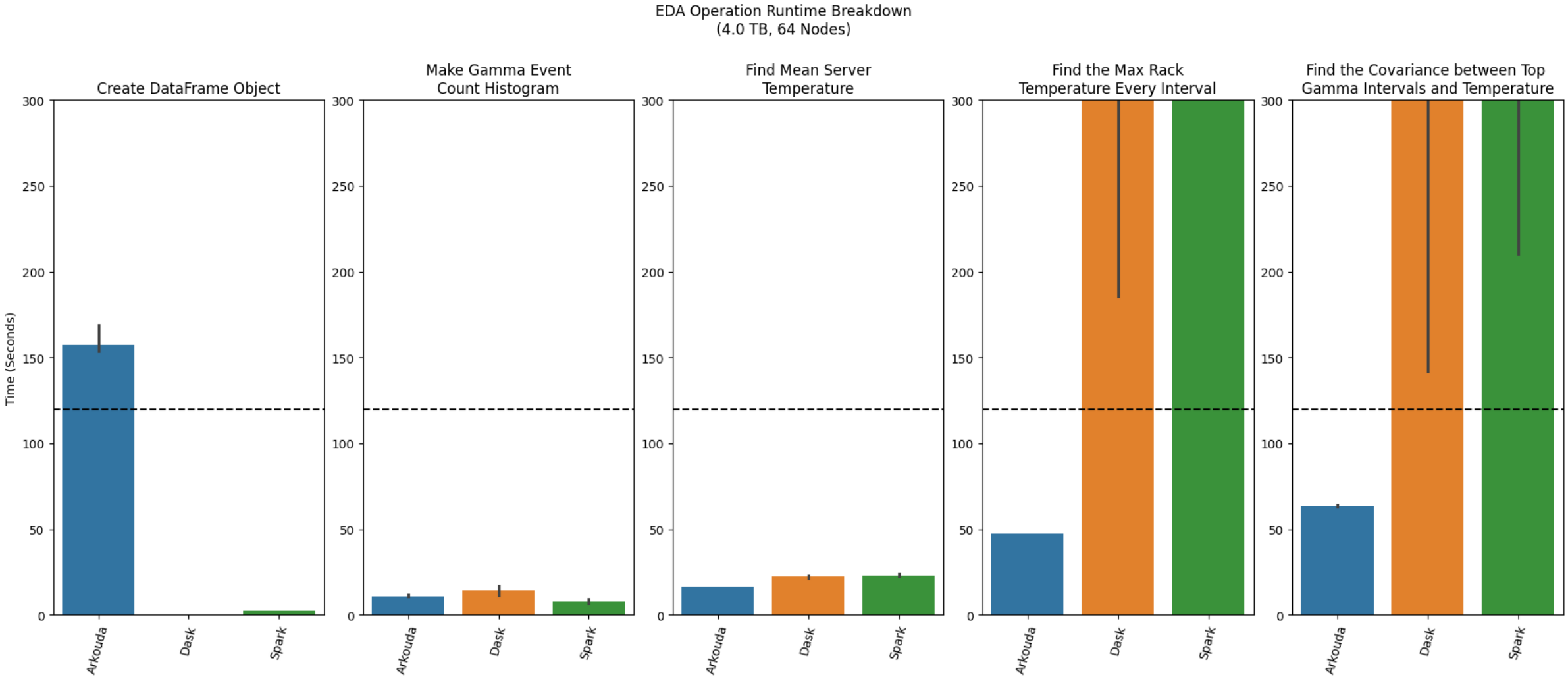
- Using the same data on a single node, Arkouda **outperformed Pandas by ~3.5x**
- Moreover, the same script shows **promising weak scaling** enabling **much larger data** to be analyzed



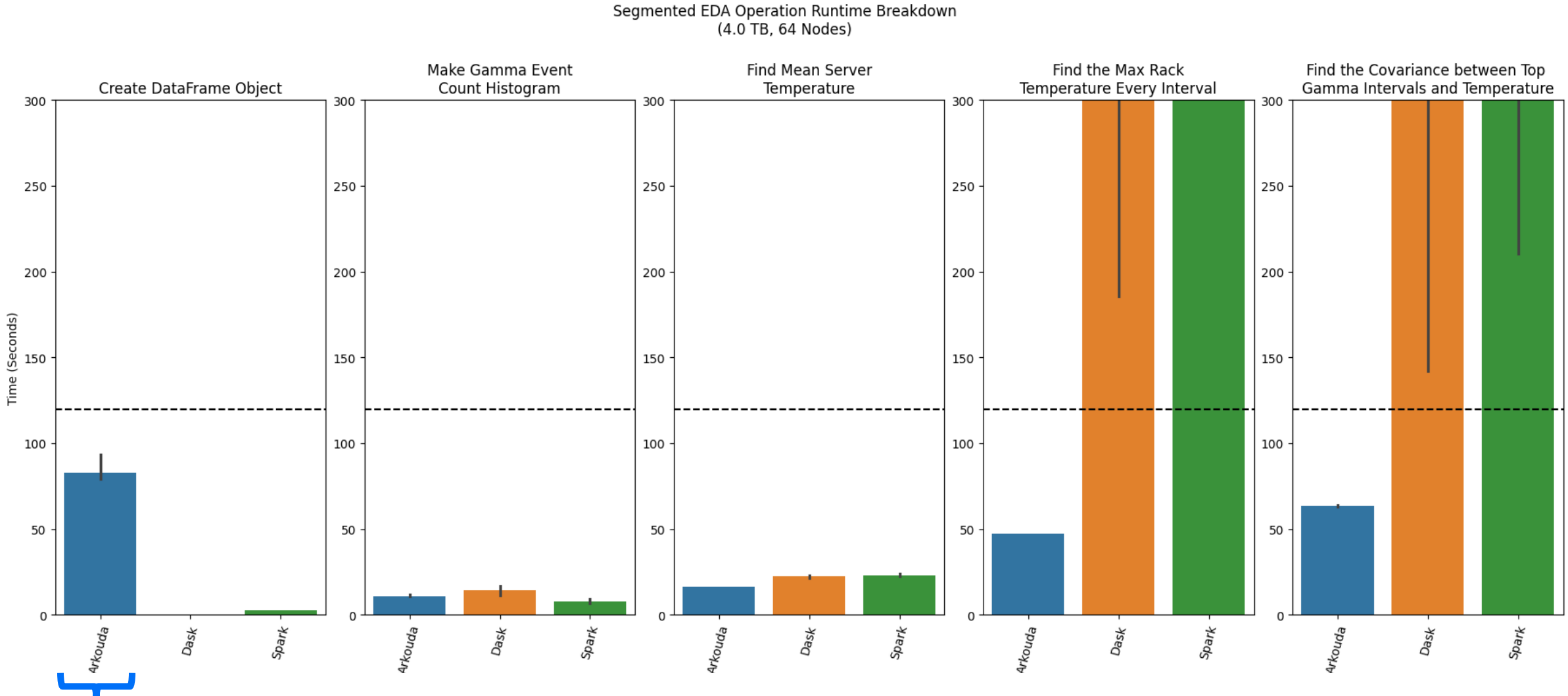
Arkouda/Dask/Spark Comparison



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

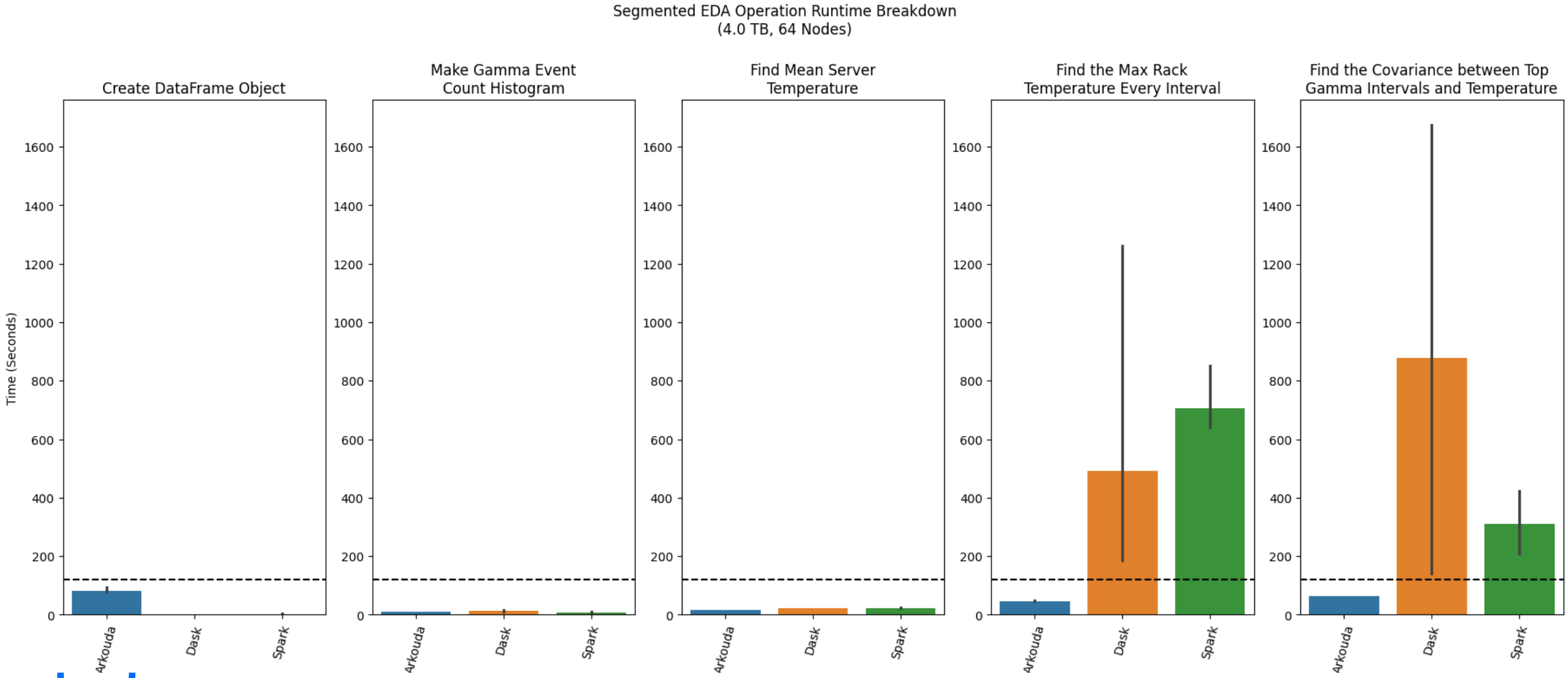


Arkouda/Dask/Spark Comparison: w/ Parquet Improvements



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

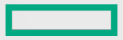
Arkouda/Dask/Spark Comparison: Zoomed out



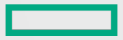
This specific bar has been updated to reflect recent improvements to Arkouda's Parquet IO

Arkouda Demo

Jade Abraham



Extending Arkouda



Extending Arkouda

After the initial NumPy/Pandas features, Arkouda added support for extending its feature set:

- new operations in the client and/or server
- new data types

Using this capability, new features have been added for:

- Multidimensional arrays
- The Python Array API / XArrays
- Zarr I/O
- SciPy operations
- Sparse matrices and sparse matrix-matrix multiplication
- Graph analytics

Graph Analytics in Arkouda (by NJIT)

Motivation: Interactive, massive-scale graph analytics are useful for:

...finding communities in co-authorship & citation graphs

...categorizing NetFlow packets as malicious vs. benign

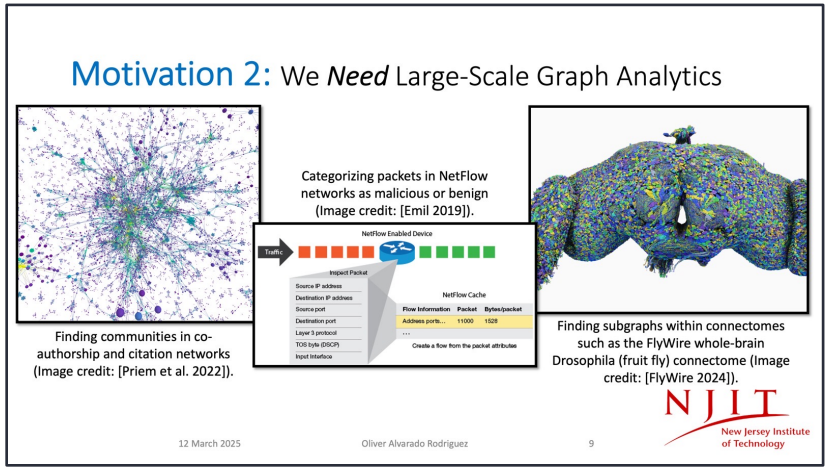
...finding subgraphs within neuroscience connectomes

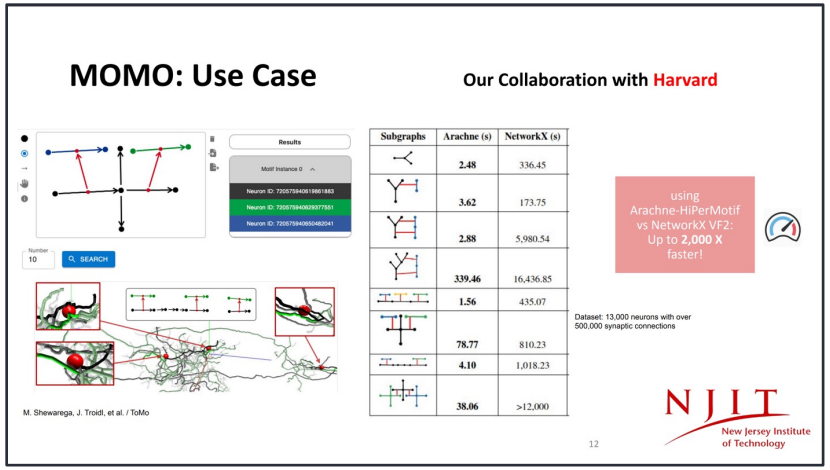
Arachne: an Arkouda extension supporting graph computations

- performance competes with or beats leading approaches
- **representations:** vertex- and edge-centric property graphs, ...
- **algorithms:** BFS, triangle counting, connected components, ...
  - subgraph isomorphism via a novel parallel algorithm and visualizer

For more information:

- *On the Design of a Framework for Large-Scale Exploratory Graph Analytics*, Oliver Andres Alvarado Rodriguez, NJIT Ph.D. dissertation, May 2025
- *HiPerMotif: Novel Parallel Subgraph Isomorphism in Large-Scale Property Graphs*, Mohammad Dindoost et al., ChapelCon '25, October 2025

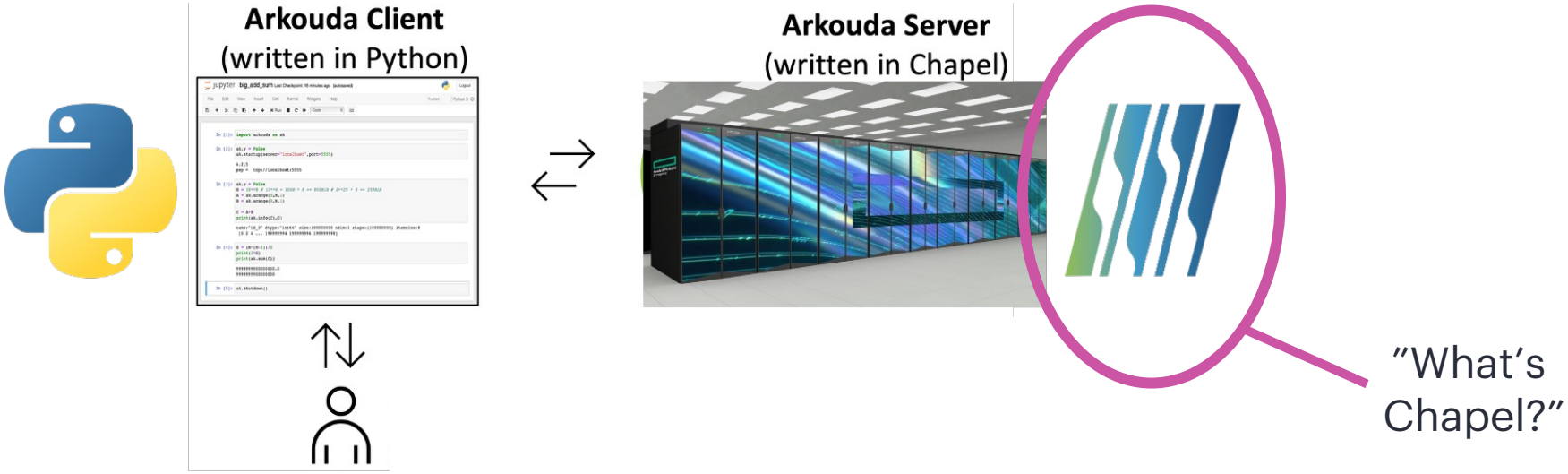




[slide images borrowed from the talks above with the authors' permission]

Key Properties of Arkouda

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



What is Chapel?

Chapel: A modern parallel programming language

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

Goals:

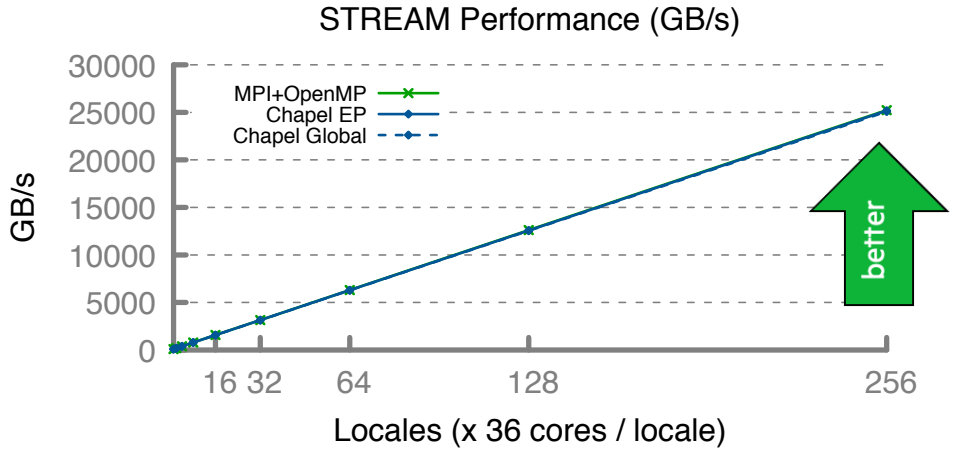
- Support general parallel programming
- Make parallel programming at scale far more productive

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 );
```

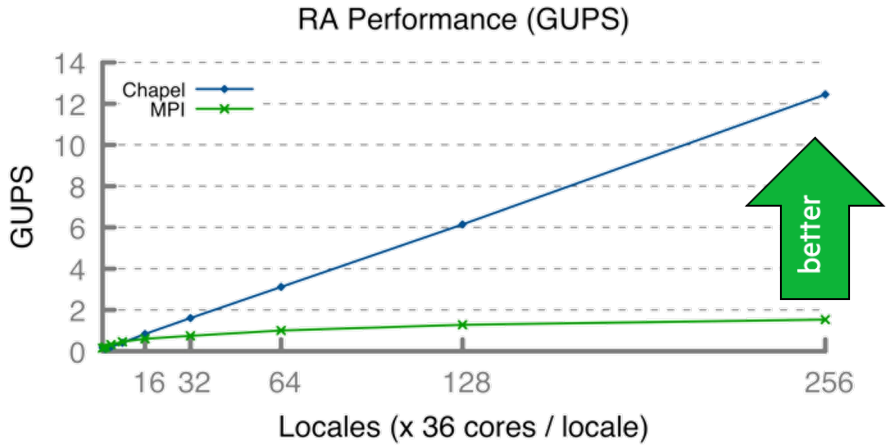
```
use BlockDist;

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 < recvUpdates; j ++) {
          inmsg = LocalRecvBuffer[bufferBase+j];
          LocalOffset = (inmsg &
                        tparams.
          HPCC_Table[LocalOffset
        }
      } else if (status.MPI_TAG
        /* we got a done message. Thanks
        NumberReceiving--;
      } else {
        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 ++) {
      inmsg = LocalRecvBuffer[bufferBase+j];
```

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



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

Chapel

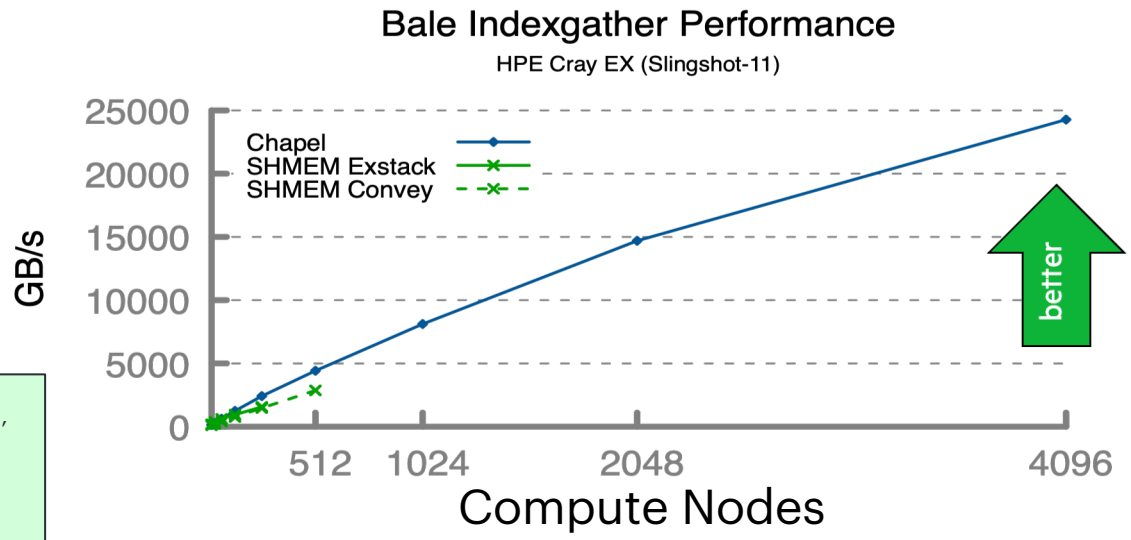
```
forall (d, i) in zip(Dst, Inds) do
  d = Src[i];
```

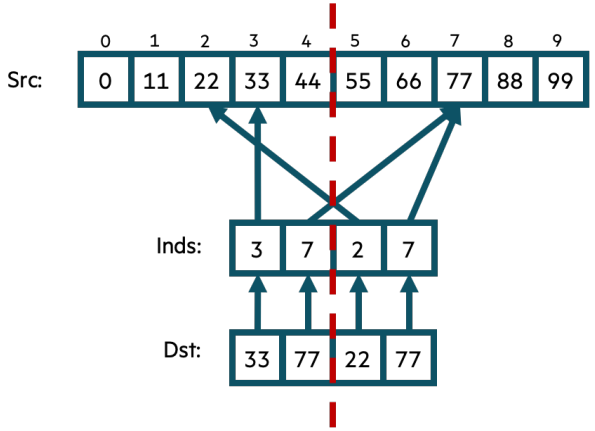
SHMEM (Exstack version)

```
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)

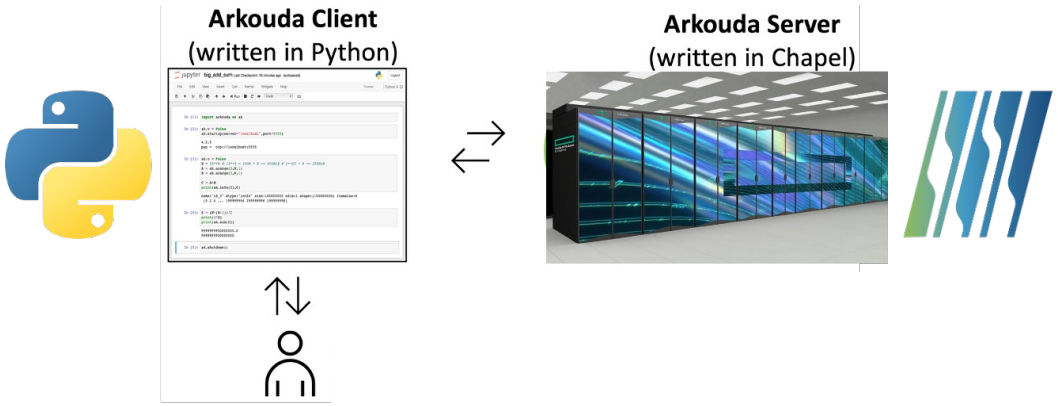
```
i = 0;
while (more = convey_advance(requests, (i == l_num_req)),
       more | convey_advance(replies, !more)) {

  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;
}
```





Why was Arkouda written in Chapel?

— **productivity**, readability, writability
  • Python-level 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 C, C++, Fortran libraries




Chapel Language Blog

About Chapel Website Featured Series Tags Authors All Posts

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

1. Who are you?
2. What do you do? What problems are you trying to solve?
3. How does Chapel help you with these problems?
4. What initially drew you to Chapel?

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?

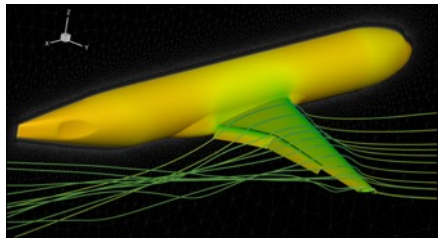
My name is Bill Reus, and I live near Annapolis, MD and the beautiful Chesapeake Bay. I am

"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."

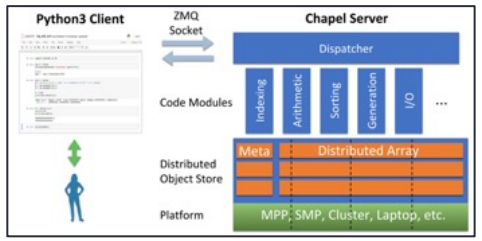
...

"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."

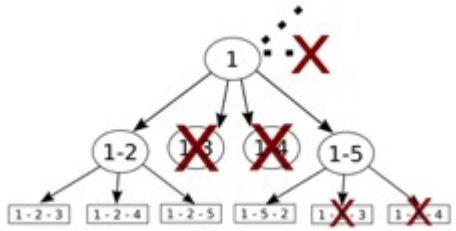
Applications of Chapel



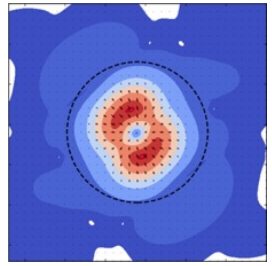
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



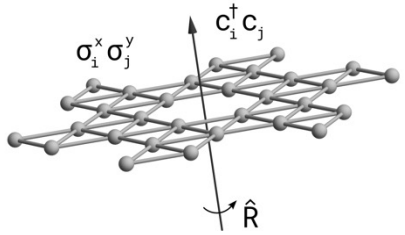
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



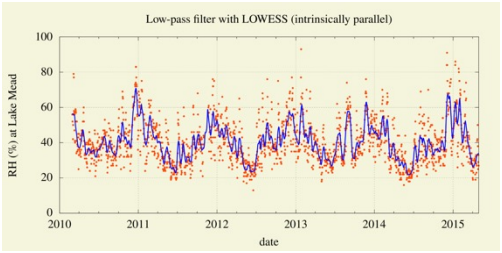
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

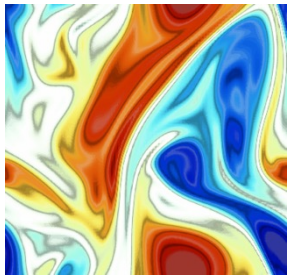


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

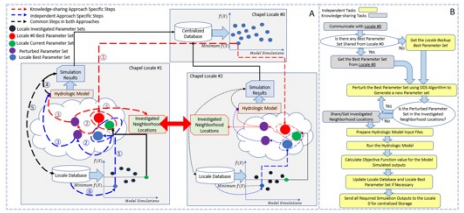


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

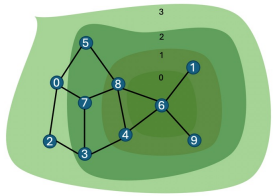
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



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

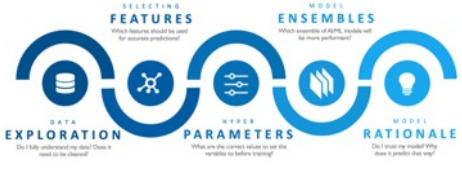


Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



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 user experiences in the “*7 Questions with Chapel Users*“ interview 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

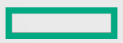
7 Questions for Marjan Asgari: Optimizing Hydrological Models with Chapel

Posted on September 15, 2025.

Tags: User Experiences | Interviews | Earth Sciences

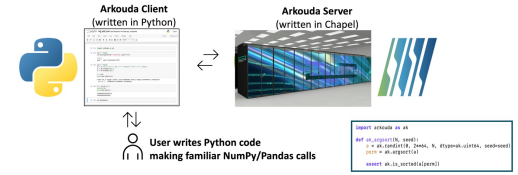
By: Engin Kayraklioglu, Brad Chamberlain

Wrap-up

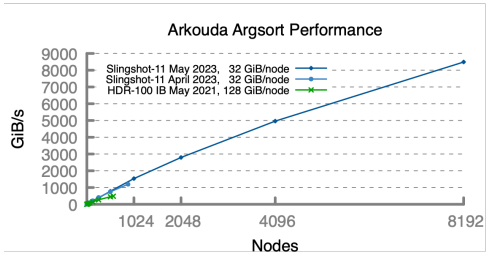


Summary

Arkouda is an open-source Python Library for driving HPC systems from Python

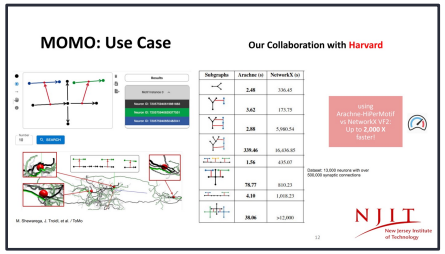


Arkouda operations have demonstrated scalability and interactivity



Arkouda's framework is extensible

- initial features focused on NumPy/Pandas APIs
- more recent work has focused on graph analytics, sparse linear algebra, SciPy, ...



Arkouda is written in Chapel for productivity, portability, and performance

CHAPEL

If you or your colleagues would like an interactive intro to Arkouda or Chapel, let us know!

What's Next?

— Continue expanding Arkouda's supported operations
- NumPy feature completeness in support of Pandas and SciPy
- Sparse linear algebra benchmarking and optimization
- …

— GPUs: Chapel supports GPU computing but, to date, Arkouda operations haven't used them
- **Big Q:** what motivating workflows and operations would benefit?

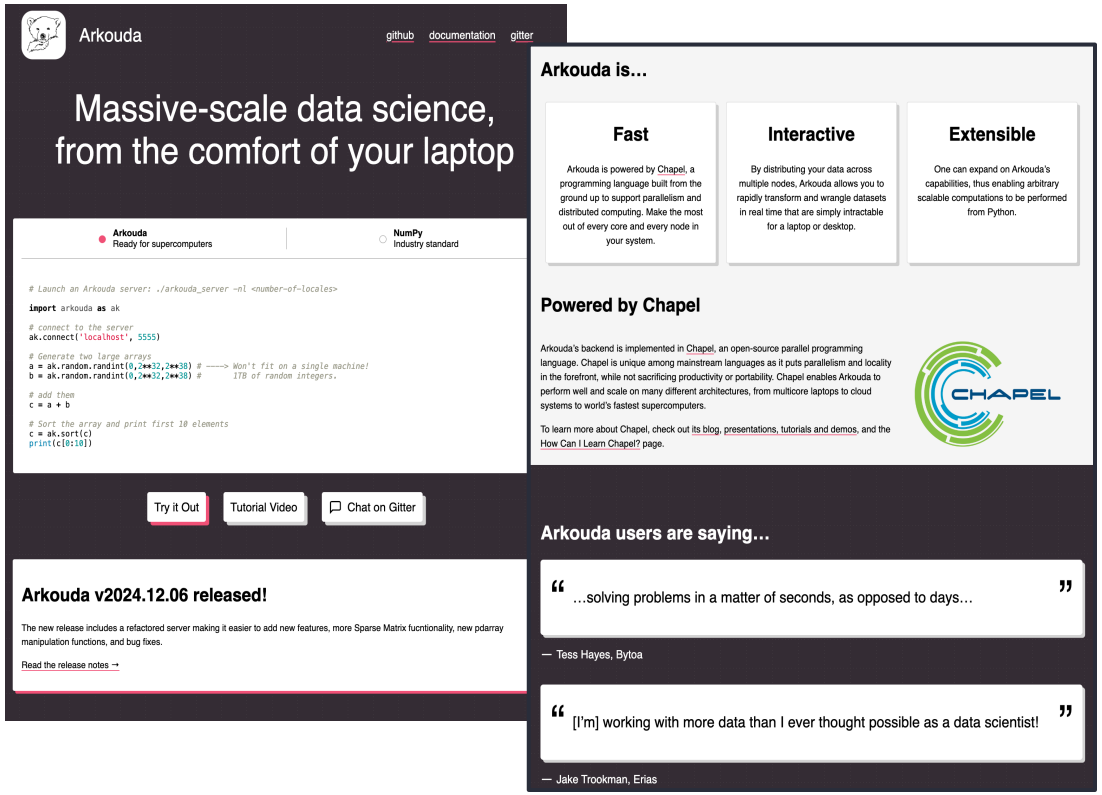
— Improve Arkouda's flexibility
- E.g., add the ability to extend the Arkouda server's capabilities dynamically

— Project Honeycomb: Next-generation version of Arkouda
- **Goals:** improved modularity and extensibility, language-agnostic, agentic-capable UI

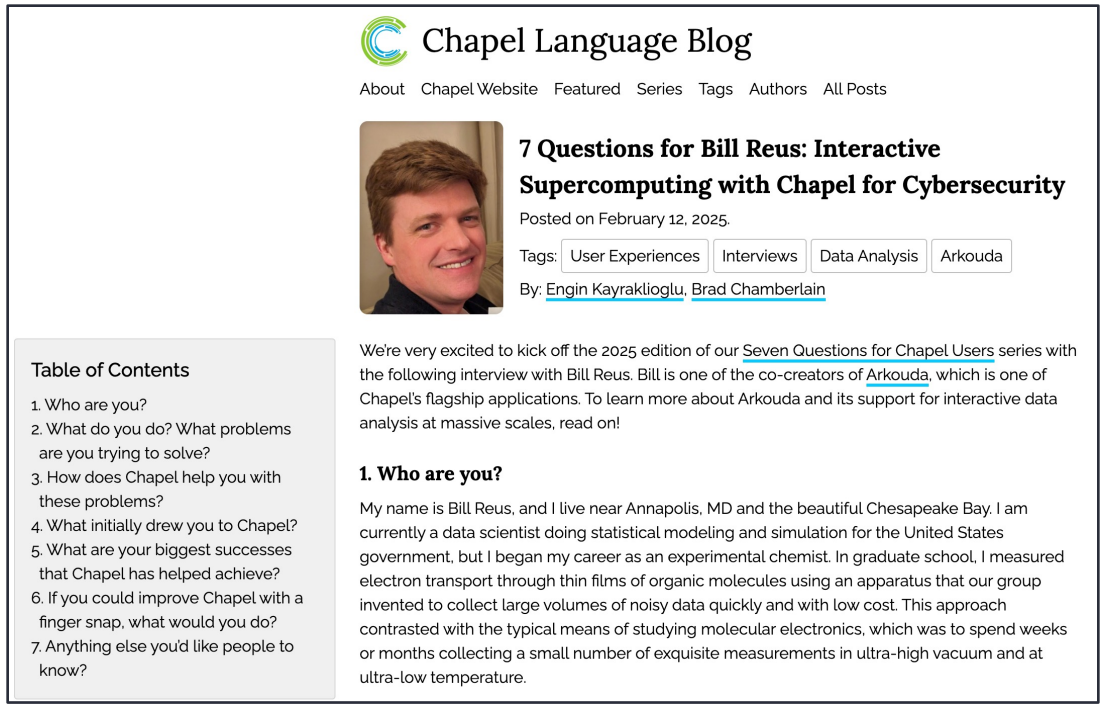
— Grow Arkouda's open-source community of users and developers

For More Information on Arkouda

Arkouda website:



Interview with founding co-developer, Bill Reus:



Recent Talk: *Arkouda Bulletin: A Year of Progress in Exploratory Data Analytics at Scale*,
Amanda Potts and Engin Kayraklioglu, ChapelCon '25, October 2025

The Advanced Programming Team at HPE

Ways to engage with the Chapel (and Arkouda) Communities

Synchronous Community Events

- Project Meetings, weekly
- Deep Dive / Demo Sessions, weekly timeslot
- Chapel Teaching Meet-up, monthly
- ChapelCon (formerly CHIUW), annually

Asynchrounous Communications

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

Social Media

FOLLOW US

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

Discussion Forums

GET IN TOUCH

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

Ways to Use Chapel

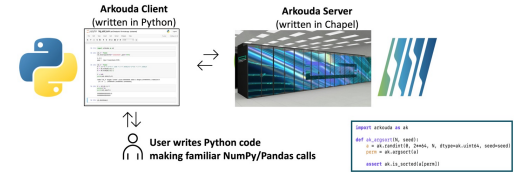
GET STARTED

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

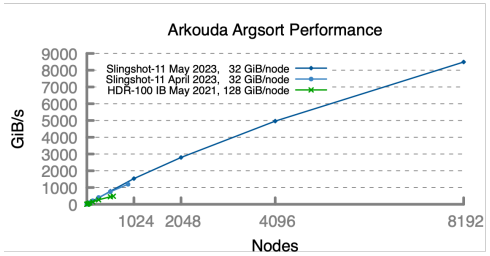
(from the footer of chapel-lang.org)

Summary

Arkouda is an open-source Python Library for driving HPC systems from Python

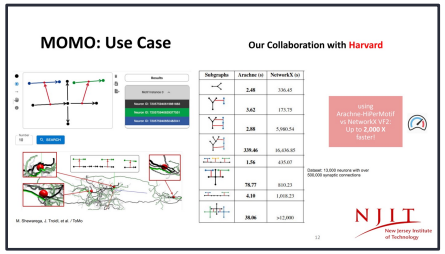


Arkouda operations have demonstrated scalability and interactivity



Arkouda's framework is extensible

- initial features focused on NumPy/Pandas APIs
- more recent work has focused on graph analytics, sparse linear algebra, SciPy, ...



Arkouda is written in Chapel for productivity, portability, and performance

CHAPEL

If you or your colleagues would like an interactive intro to Arkouda or Chapel, let us know!

Thank You

@ChapelLanguage