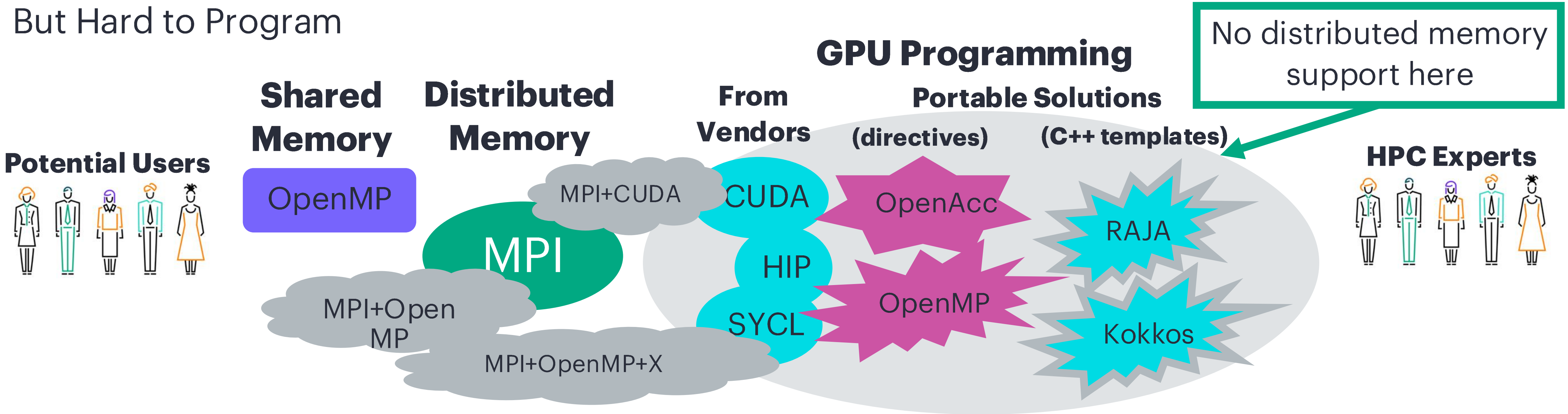
Productive Parallel Programming with Chapel and Arkouda

Jade Abraham (@jabraham17), Advanced Programming Team, HPE

FOSDEM 2026
February 1, 2026

Parallel Systems are Easy to Find...

But Hard to Program



All are effective, powerful, essential, and tested technologies!

- ...but
  - all of these are based on C/C++/Fortran
  - some paradigms haven't changed in decades (i.e. MPI)
  - mixing parallel hardware requires multiple frameworks
  - higher-level abstractions often exchange performance and control for ease-of-use

As a result, HPC has a high barrier to entry

An Alternative for Productive Parallel Programming

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

Productive Parallel Programming

- Imagine a language that is as...
  - ...**readable and writeable** as Python
  - ...**fast** as Fortran / C / C++ / Rust
  - ...**scalable** as MPI / SHMEM
  - ...**GPU-ready** as CUDA / HIP / OpenMP / Kokkos / OpenCL / OpenACC / ...
  - ...**portable** as C
  - ...**fun** as [your favorite language]

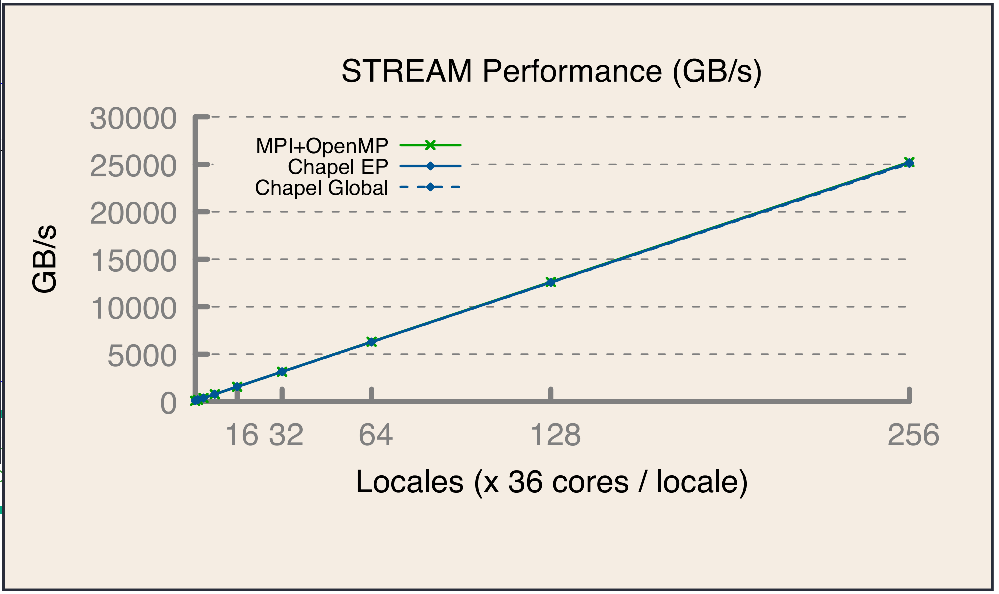
This is the motivation for Chapel

Stream Triad: C + MPI + OpenMP vs Chapel

```
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;
```

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

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

```
  if (!a || !b || !c) {
...File, "Failed to allocate memory (%d).\n", VectorSize );
...ile );

    a[j] = b[j]+sc...

  HPCC_free(c);
  HPCC_free(b);
  HPCC_free(a);

  return 0;
}
```



HPCC RA: C + MPI vs Chapel

```
/* 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;
```

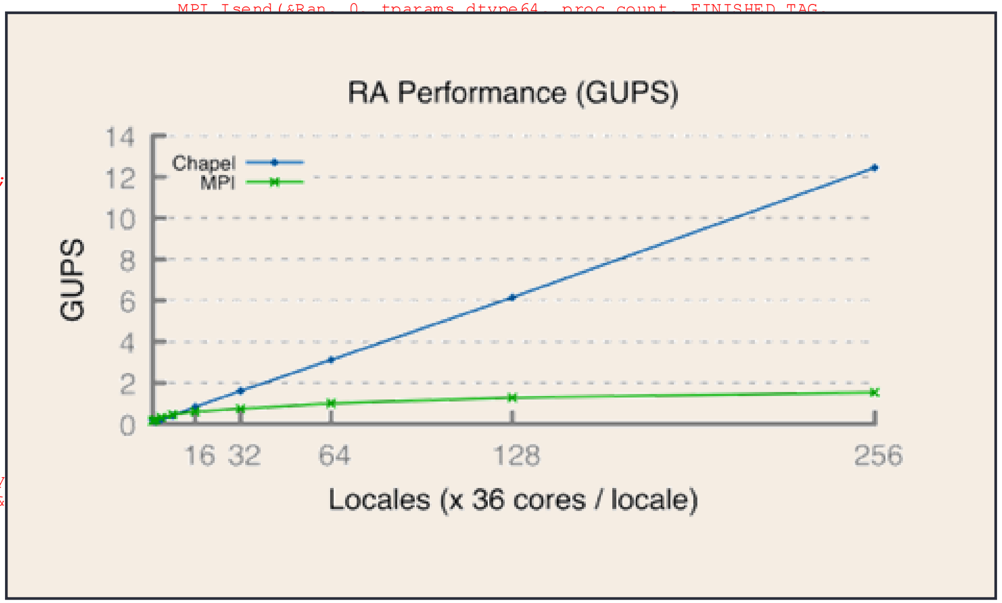
```
MPI_STATUS_IGNORE);
LocalSendBuffer, localBufferSize,
es);
eUpdates, tparams.dtype64, (int)pe,
M_WORLD, &outreq);
< tparams.NumProcs ; ++proc_count) {
c) { tparams.finish_req[tparams.MyProc] =
       MPI_REQUEST_NULL; continue; }
/* send garbage - who cares, no one will look at it */
```

```
/* 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;
 *    }
 */
```

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

```
      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;
```

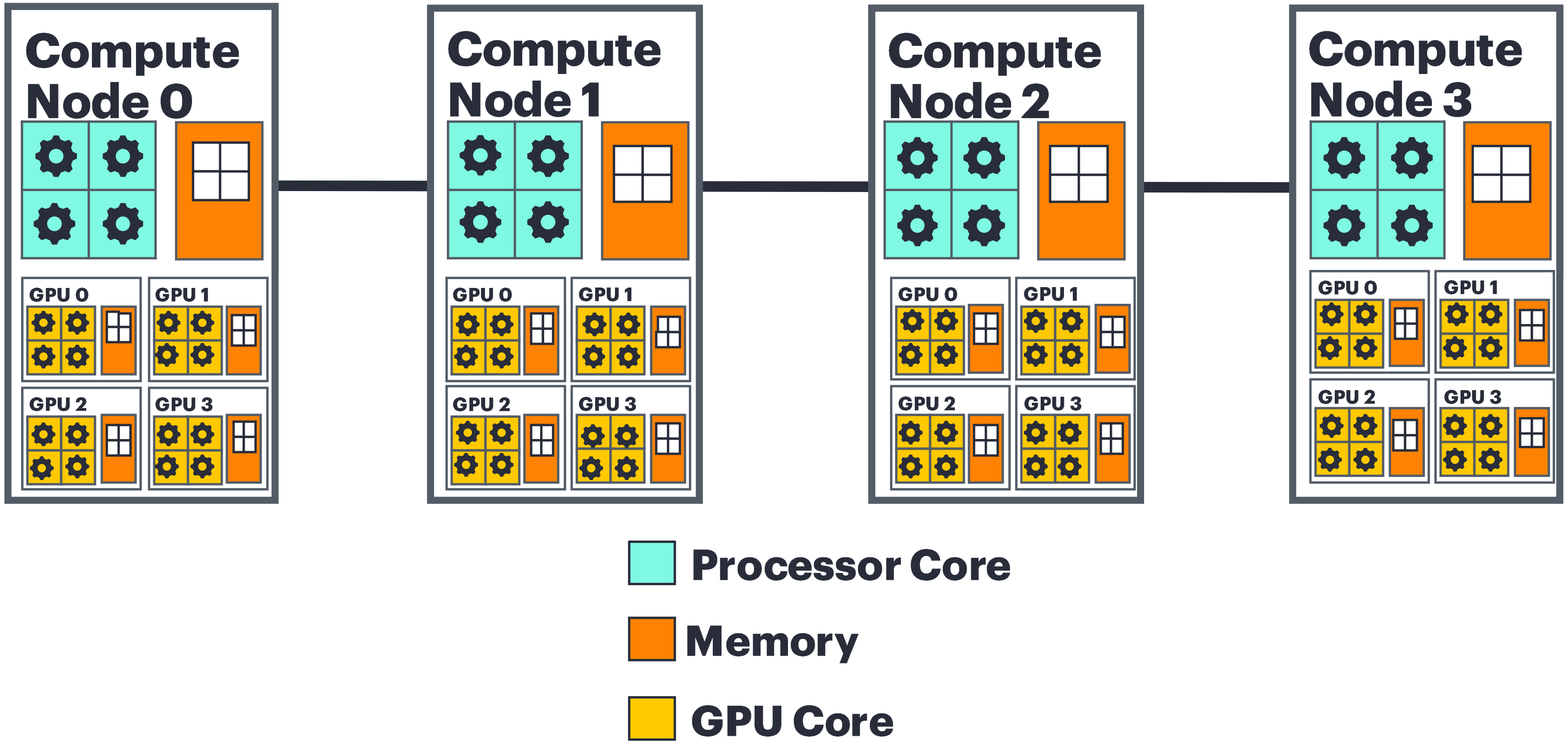
```
      } else if (status.MPI_TAG == FINISHED_TAG) {
        /* we got a done message.  Thanks for playing... */
        NumberReceiving--;
      } else {
        MPI_Abort( MPI_COMM_WORLD, -1 );
      }
      MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dty
                MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &
    }
  } while (have_done && NumberReceiving > 0);
```



Key Language Features

Key Concerns for Scalable Productive Parallel Computing

1. **parallelism:** What computational tasks should run simultaneously?
2. **locality:** Where should tasks run? Where should data be allocated?



Basic Features For Locality

```
writeln("Hello from locale ", here.id);

var A: [1..2, 1..2] real;

for loc in Locales {
  on loc {
    var B = A;
    for gpu in loc.gpus {
      on gpu {
        var C = B;
      }
    }
  }
}
```

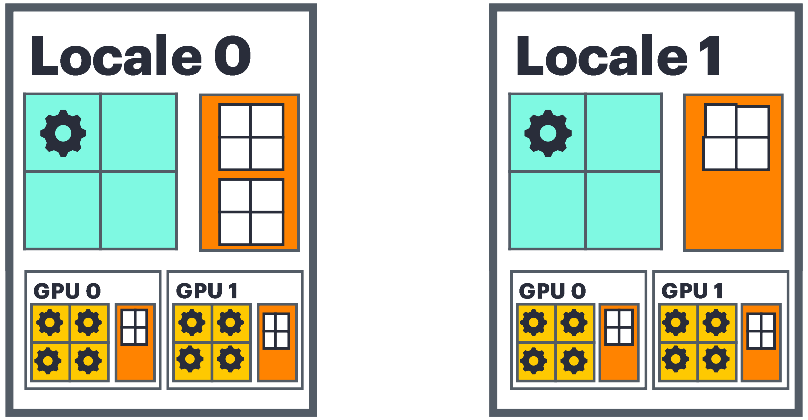
All Chapel programs start with one task on locale 0

Variables are stored using the current task's local memory

A serial loop for each of the program's locales

on-clauses move the computation to the target locale

Remote variables can be accessed directly



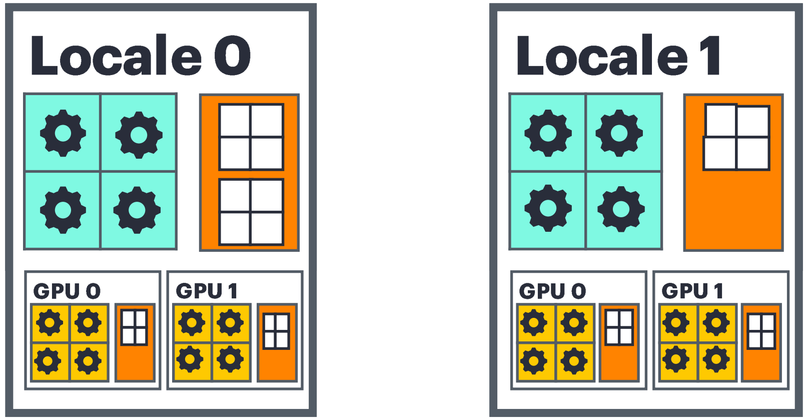
This is distributed *serial* computation

Locality + Parallelism

```
writeln("Hello from locale ", here.id);

var A: [1..2, 1..2] real;

coforall loc in Locales {
  on loc do cobegin {
    var B = A;
    coforall gpu in loc.gpus {
      on gpu {
        var C = AB;
      }
    }
  }
}
```

The coforall loop creates a parallel task per **iteration**

The cobegin statement creates a parallel task per **child statement**



This is distributed *parallel* computation

One Loop To Rule Them All

```
proc increment(Arr) {
  var Res: Arr.type;
  forall i in Arr.domain {
    Res[i] = Arr[i] + 1;
  }
  return Res;
}
```

equivalently

```
proc increment(Arr) {
  return Arr + 1;
}
```

The forall loop will invoke the parallel iterator for its iterand expression, in this case the indices of Arr

```
var myLocalArr: [1..10] int;
increment(myLocalArr);
```

Single node parallel computation

```
use BlockDist;
var myDistributedArr = blockDist.createArray({1..10}, int);
increment(myDistributedArr);
```

Distributed parallel computation

```
on here.gpu[0] {
  var myGpuArr: [1..10] int;
  increment(myGpuArr);
}
```

Single GPU parallel computation

One Loop To Rule Them All + Interoperability

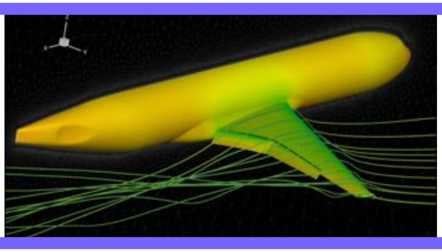
```
extern {
  #include <stdint.h>
  static int64_t increment_c(int64_t X) { return X + 1; }
}

proc increment(Arr) {
  var Res: Arr.type;
  forall i in Arr.domain {
    Res[i] = increment_c(Arr[i]);
  }
  return Res;
}

var myLocalArr: [1..10] int;
increment(myLocalArr);

use BlockDist;
var myDistributedArr = blockDist.createArray({1..10}, int);
increment(myDistributedArr);
```

It doesn’t even have to be "external”!

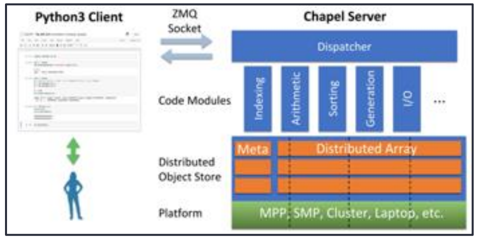
Instead of Chapel code, call an external procedure

Applications

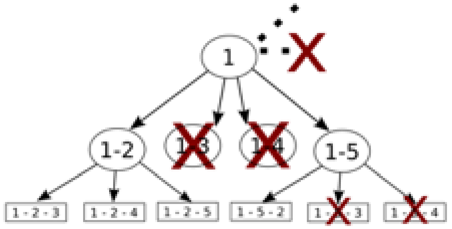
Applications of Chapel



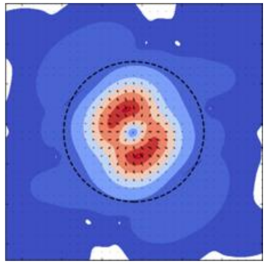
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



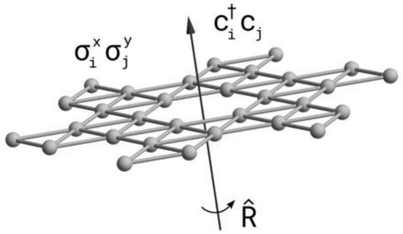
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



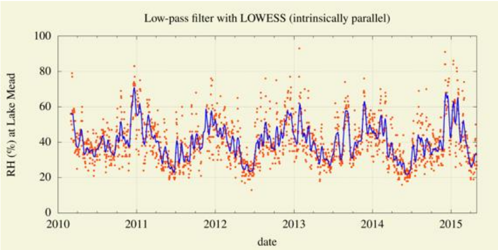
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

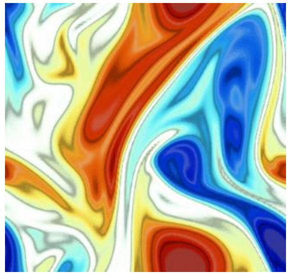


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

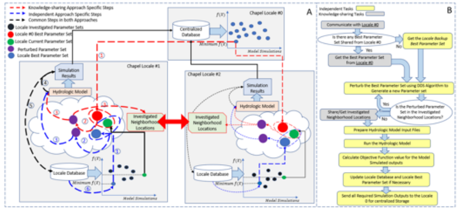


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

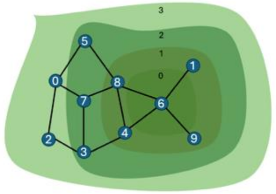
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



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

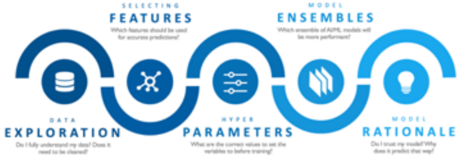


Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



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

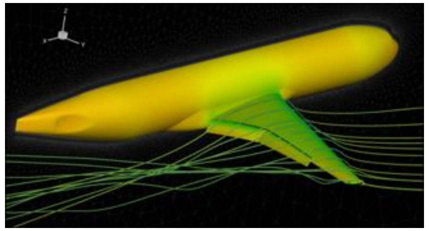
Modeling Ocean Carbon Dioxide Removal
Scott Bachman Brandon Neth, et al.
[C]Worthy



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

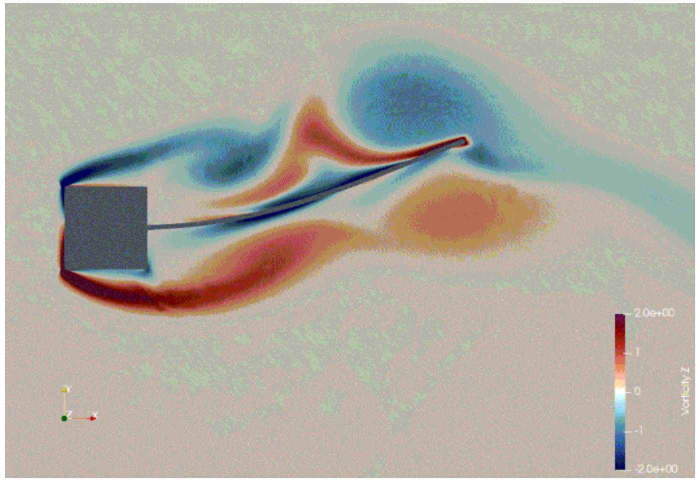
[images provided by their respective teams and used with permission]

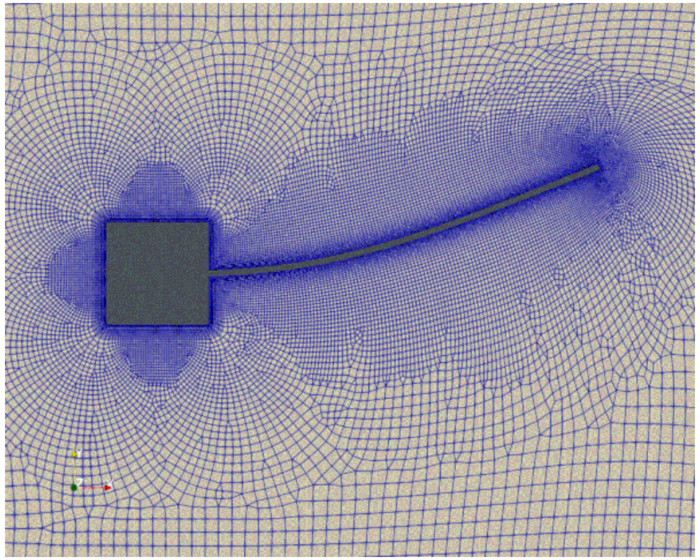
Applications of Chapel: CHAMPS



What is it?

- 3D unstructured CFD framework for airplane simulation
- ~100+k lines of Chapel written since 2019





Who wrote it?

- Professor Éric Laurendeau’s students + postdocs at Polytechnique Montreal

POLYTECHNIQUE MONTRÉAL

Why Chapel?

- performance and scalability competitive with MPI + C++
- students found it far more productive to use
- enabled them to compete with more established CFD centers

[images provided by the CHAMPS team and used with permission]

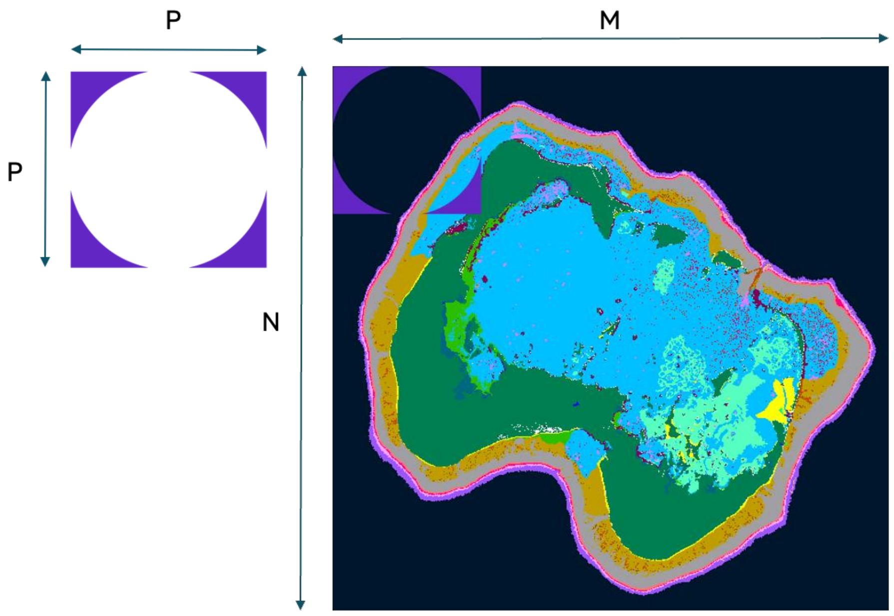
Applications of Chapel: Biodiversity

What is it?

- Measures coral reef diversity using high-res satellite image analysis
- ~230 lines of Chapel code written in late 2022

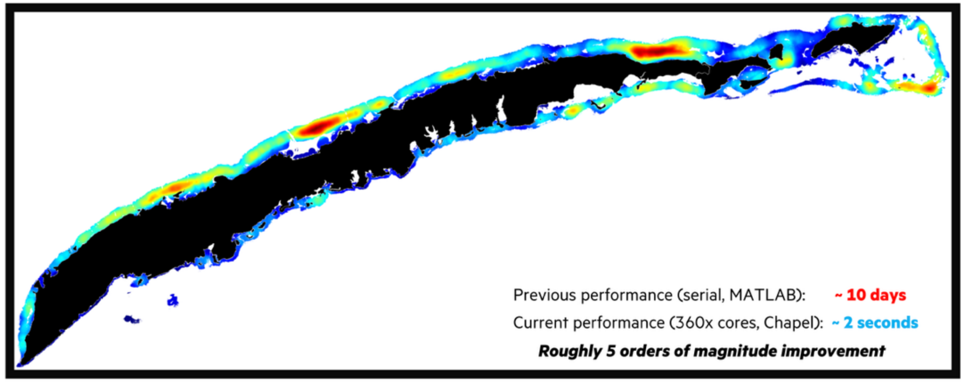
Who wrote it?

- Scott Bachman, NCAR/[C]Worthy
  - with Rebecca Green, Helen Fox, Coral Reef Alliance



Why Chapel?

- easy transition from Matlab, which they had been using
- massive performance improvement:
  previous ~10-day run finished in ~2 seconds using 360 cores
- enabled unanticipated algorithmic improvements
  - from $O(M \cdot N \cdot P)$ habitat diversity to $O(M \cdot N \cdot P^3)$ spectral diversity
  - Added another ~90 lines of code to make it GPU-enabled
  - ~4-week desktop run → ~20 minutes on 20 nodes / 512 GPUs



[images provided by Scott Bachman from his CHIUW 2023 talk and used with permission]

Why Chapel?

Get out of the way of science

- Let scientists express their computations succinctly
- Python and other languages are already good at this
  - But to get performance and scalability, you tend to give up the nice abstractions

Easier to write, easier to read, easier to maintain

- Lowers the barrier to entry for new contributors
- You don’t need to be a MPI savant to get good distributed performance
- You don’t need to be a CUDA wizard to use GPUs efficiently

Applications: Arkouda

Data Science In Python at scale?

Motivation: Imagine you work with…

…Python-based data scientists

…HPC-scale data science problems to solve

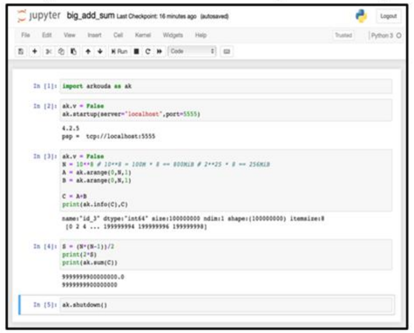
…access to HPC systems

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

What is Arkouda?

Q: "What is Arkouda?"

Arkouda Client
(written in Python)



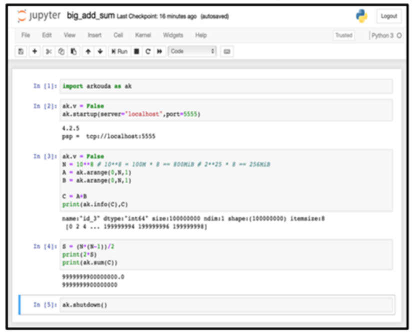
```python
import arkouda as ak

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

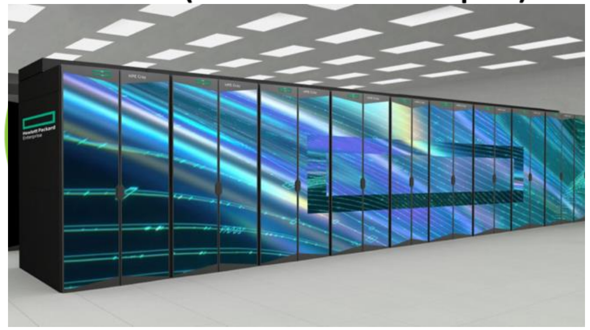
What is Arkouda?

Q: "What is Arkouda?"

Arkouda Client
(written in Python)



Arkouda Server
(written in Chapel)



User writes Python code making familiar NumPy/Pandas calls

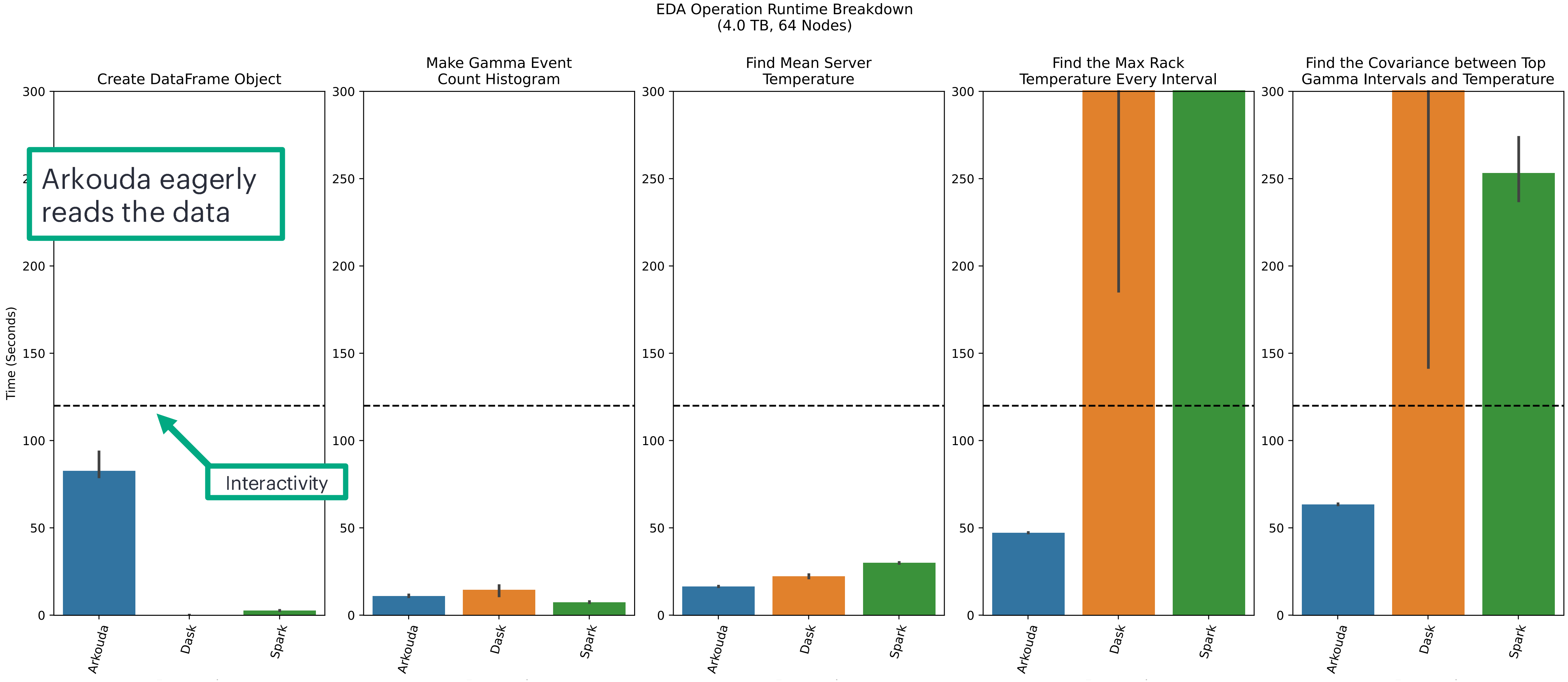
```
import arkouda as ak

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"

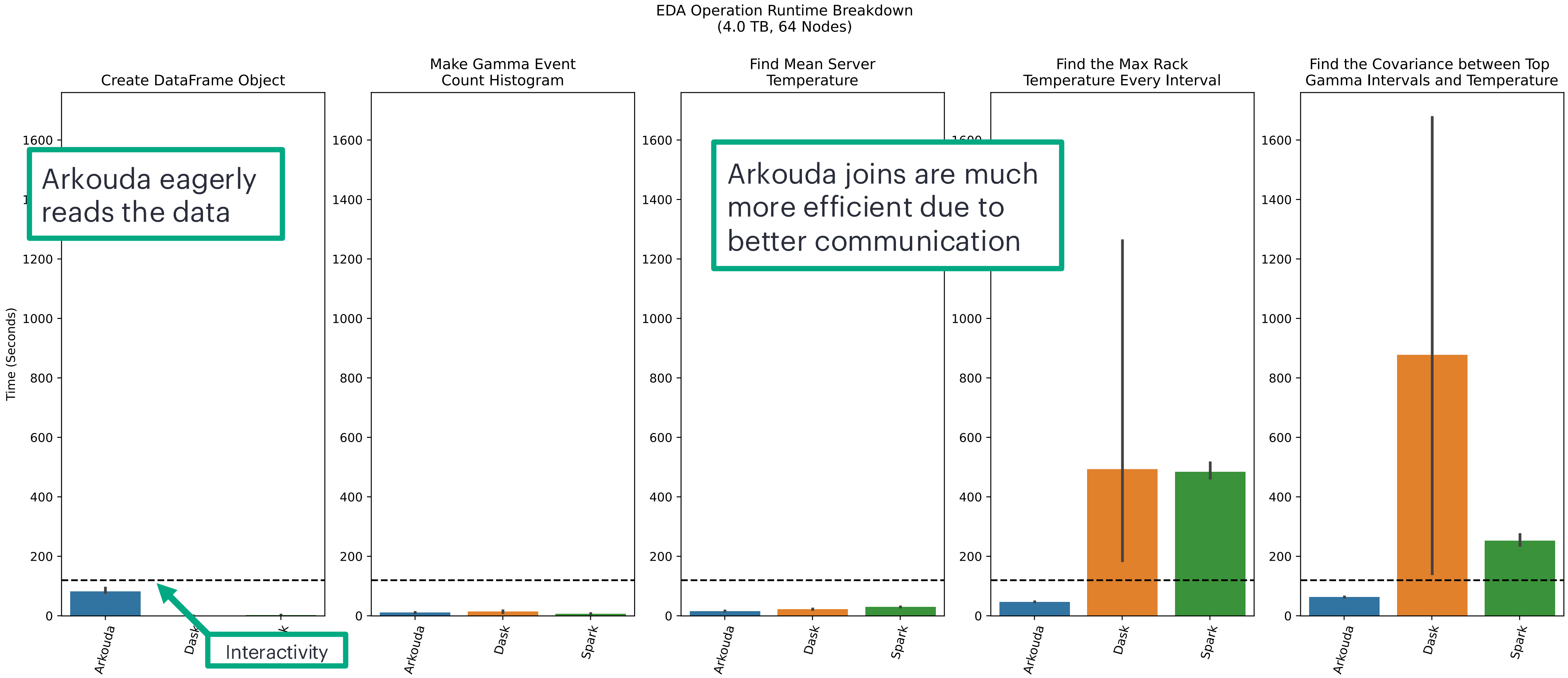
A2: "A framework for driving supercomputers interactively from Python"

Chapel/Arkouda Performance

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



Arkouda/Dask/Spark Comparison: Zoomed out



Summary

Chapel is built from the ground up for productive parallel computing at any scale

- Language features express clear and concise computation

Chapel applications can meet or beat low-level approaches while still...

- ...avoiding unnecessary boilerplate
- ...being human-readable
- ...working with existing workflows

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

Part of a series: 7 Questions for Chapel Users

https://chapel-lang.org/blog/posts/7qs-reus

“ *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.* ”

Ways to engage with the Chapel Community

Synchronous Community Events

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

Asynchronous 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)