Arkouda: Data Science at Massive Scales and Interactive Rates

Brad Chamberlain

Puget Sound Programming Python (PuPPy)

February 12, 2020

chapel_info@cray.com
chapel-lang.org
@ChapelLanguage

CRAY®
a Hewlett Packard Enterprise company

CRAY CHAPEL

Defining our Terms

“Data Science”: human-in-the-loop data analysis using familiar interfaces

- **“Familiar Interfaces:”** NumPy / Pandas operations

“Massive Scales:” dozens of terabytes of data (e.g., 30–90 TB)

“Interactive Rates:” operations complete in seconds to a few minutes

Motivation for Arkouda

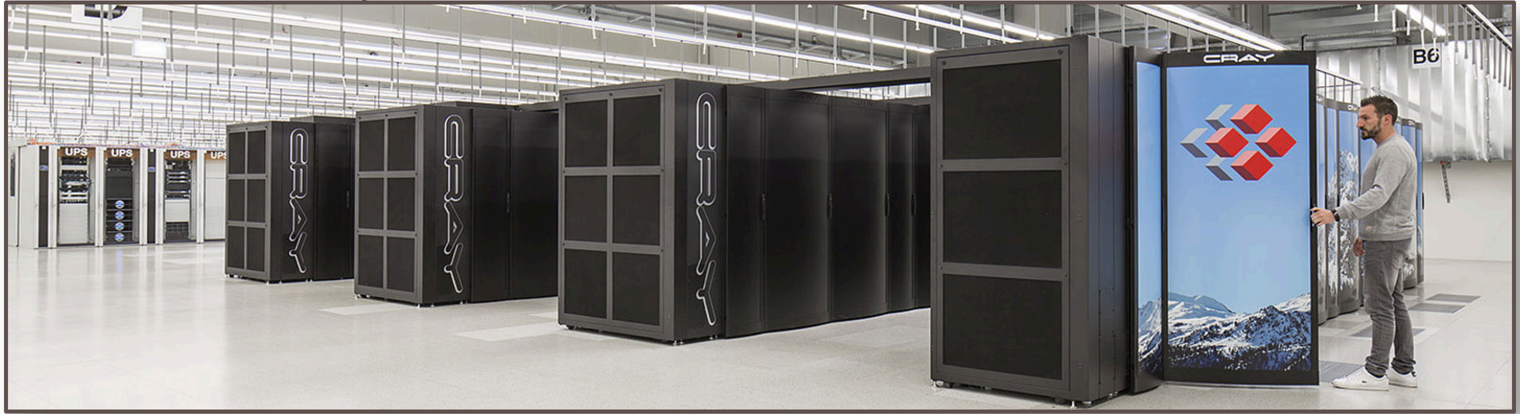
Motivation: Say you've got…

…a bunch of Python programmers

…HPC-scale problems to solve

…access to HPC systems

https://www.cscs.ch/computers/piz-daint/



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

What is Chapel?

Chapel: A modern parallel programming language

- portable & scalable
- open-source & collaborative

Goals:

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

Data Parallelism in Chapel, by example

dataParallel.chpl

```
use CyclicDist;

config const n = 1000;

var D = {1..n, 1..n} dmapped Cyclic(startIdx = (1,1)),
    A: [D] real;

forall (i,j) in D do
  A[i,j] = i + (j - 0.5)/n;

writeln(A);
```

```
prompt> chpl dataParallel.chpl
prompt> ./dataParallel --n=5 --numLocales=4
1.1 1.3 1.5 1.7 1.9
2.1 2.3 2.5 2.7 2.9
3.1 3.3 3.5 3.7 3.9
4.1 4.3 4.5 4.7 4.9
5.1 5.3 5.5 5.7 5.9
```

Recent Notable Chapel Use Cases

Yale

Simulation of Ultralight Dark Matter
Nikhil Padmanabhan et al.
Yale University

3D Computational Fluid Dynamics
Simon Bourgault-Côté, Matthieu Parenteau, et al.
École Polytechnique Montréal

Pacific Northwest NATIONAL LABORATORY

Chapel Hypergraph Library (CHGL)
Louis Jenkins, Marcin Zalewski, et al.
PNNL

Arkouda: NumPy at Scale
Mike Merrill, Bill Reus, et al.
US DOD

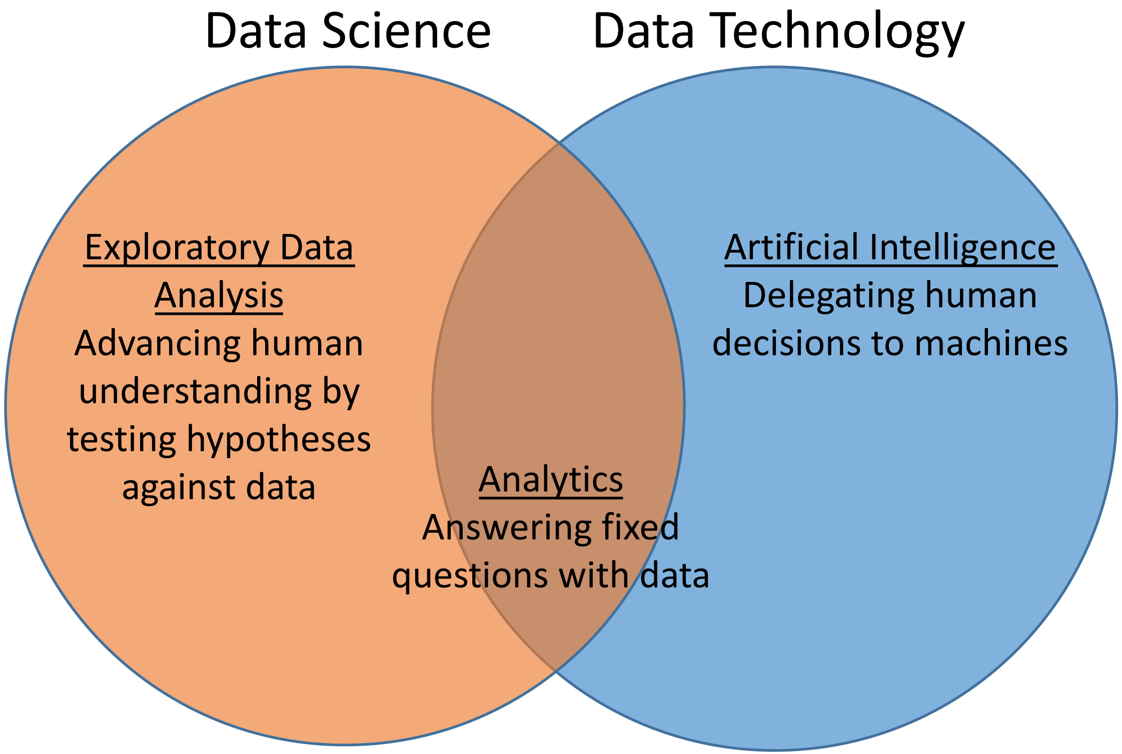
Data Science Needs Interactive Supercomputing

Dr. William Reus

US Department of Defense

presented at CLSAC 2020, October 9, 2019

"Can" Does Not Imply "Should"



Science is critical:

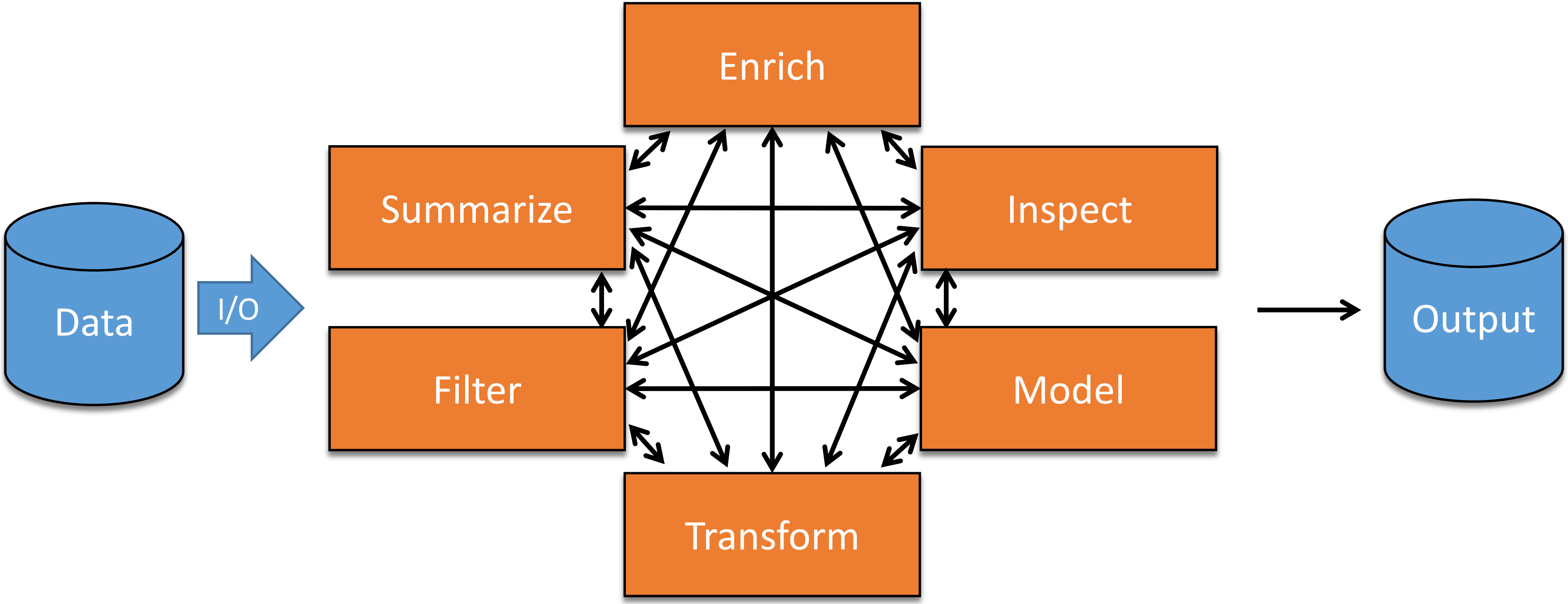
- Technology is not always the right goal
- Tech. without science will fail

And yet…

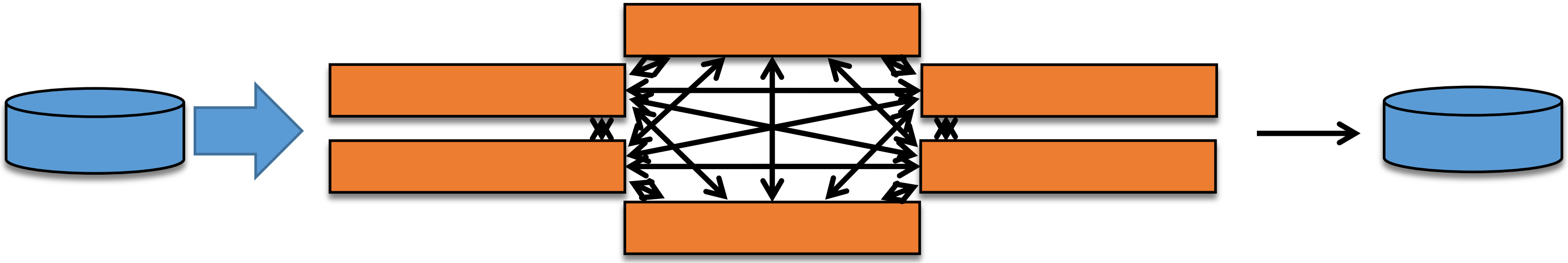
- Technology is what everyone talks about
- Large-scale tools favor tech. over science

(Data) Science is Interactive

"Hypothesis Testing"



Implications for Computing



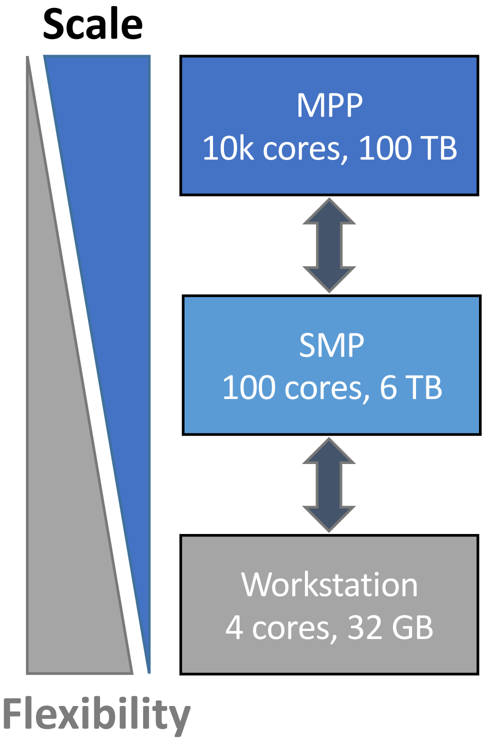
- Stay in memory
- Compute in small, reversible steps
- Enable introspection (code and state)
- Use other people's code
- Avoid boilerplate
- Maximize $\frac{t_{thinking}}{t_{thinking}+t_{coding}+t_{waiting}}$

So, basically Python…

…but fast

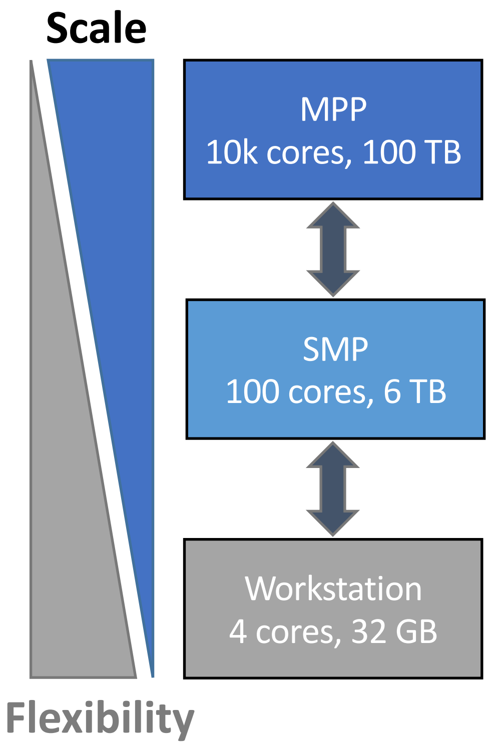
Interactive Computational Ladder

- Goal: Move seamlessly between tiers
  - Same data formats
  - Same UI (Jupyter)
  - Same APIs (NumPy/Pandas)
- Lower two tiers are easy



Interactive Computational Ladder

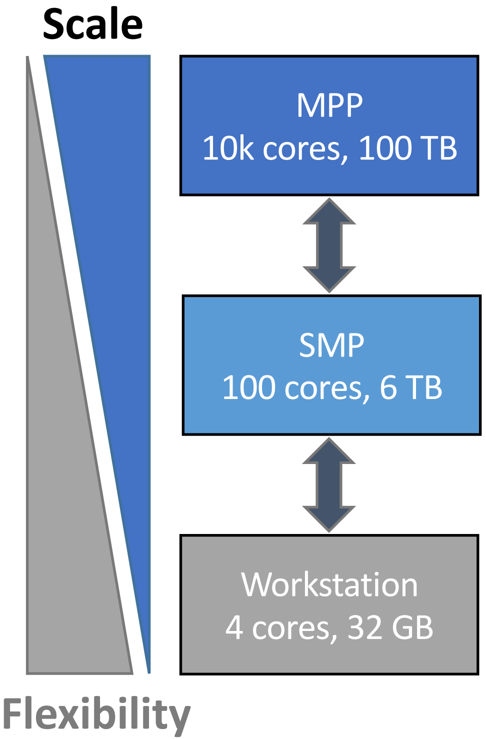
- We need the upper tier
  - Cybersecurity data >> 6 TB
- But hardware is the easy part
  - Need serious data engineering
  - Need to rethink job scheduling
  - Need an **HPC shell**



Interactive Computational Ladder

- We need the upper tier
  - Cybersecurity data >> 6 TB
- But hardware is the easy part
  - Need serious data engineering
  - Need to rethink job scheduling
  - Need an **HPC shell**

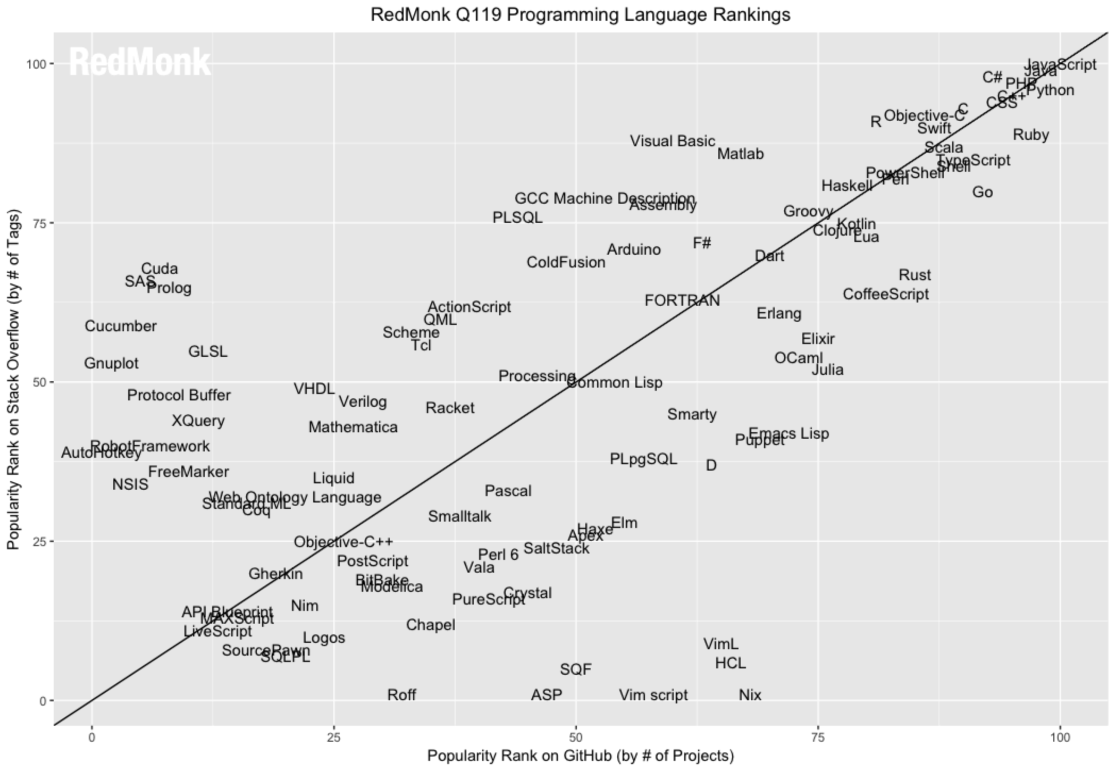
Brad: "So, basically Python…
…but fast
…and scalable"



Python Strengths

a Hewlett Packard Enterprise company

https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/



Pros:

- Hugely popular
- …

Mar 2019	Mar 2018	Change	Programming Langu		
1	1		Java		
2	2		C		
3	**4**	^	**Python**		
4	3	v	C++		
5	6	^	Visual Basic .NET		
6	5	v	C#		
7	8	^	JavaScript		
8	7	v	PHP		
9	10	^	SQL		
10	14	^^	Objective-C	1.681%	-0.09%

https://www.tiobe.com/tiobe-index/

Python Strengths

Pros:

- Hugely popular
- Extremely readable / writeable
- Massive number of libraries
- Strong community and online presence
- Supports interactive programming
- Dynamic typing (convenient!)
- …

Python Weaknesses [for HPC]

Cons:

- Weak support for parallelism and scalability
- Most performance obtained by calling into C code
- Poor support for large-scale software projects
- Dynamic typing (surprising errors at execution time!)
- …

Arkouda's Key Idea

a Hewlett Packard Enterprise company

Motivation for this effort

The Challenge: Say you've got…

…an army of Python programmers

…HPC-scale problems to solve

…access to HPC systems

How should you leverage these Python programmers to get your work done?

Python Weaknesses

Cons:

- Weak support for parallelism and scalability
- Most performance obtained by calling into C code
- Poor support for large-scale projects
- Dynamic typing (surprising errors at execution time!)
- …

Concept: Develop Python libraries that are implemented in Chapel

⇒ get performance, as with C-based libraries, but also parallelism + scalability

Even Better: use familiar interfaces (e.g., NumPy) to make it trivial for users

An HPC Shell for Data Science

Load Terabytes of data…

… into a familiar, interactive UI …

… where standard data science operations …

… execute within the human thought loop …

… and interoperate with optimized libraries.

Arkouda

Load Terabytes of data…

… into a familiar, interactive UI …

… where standard data science operations …

… execute within the human thought loop …

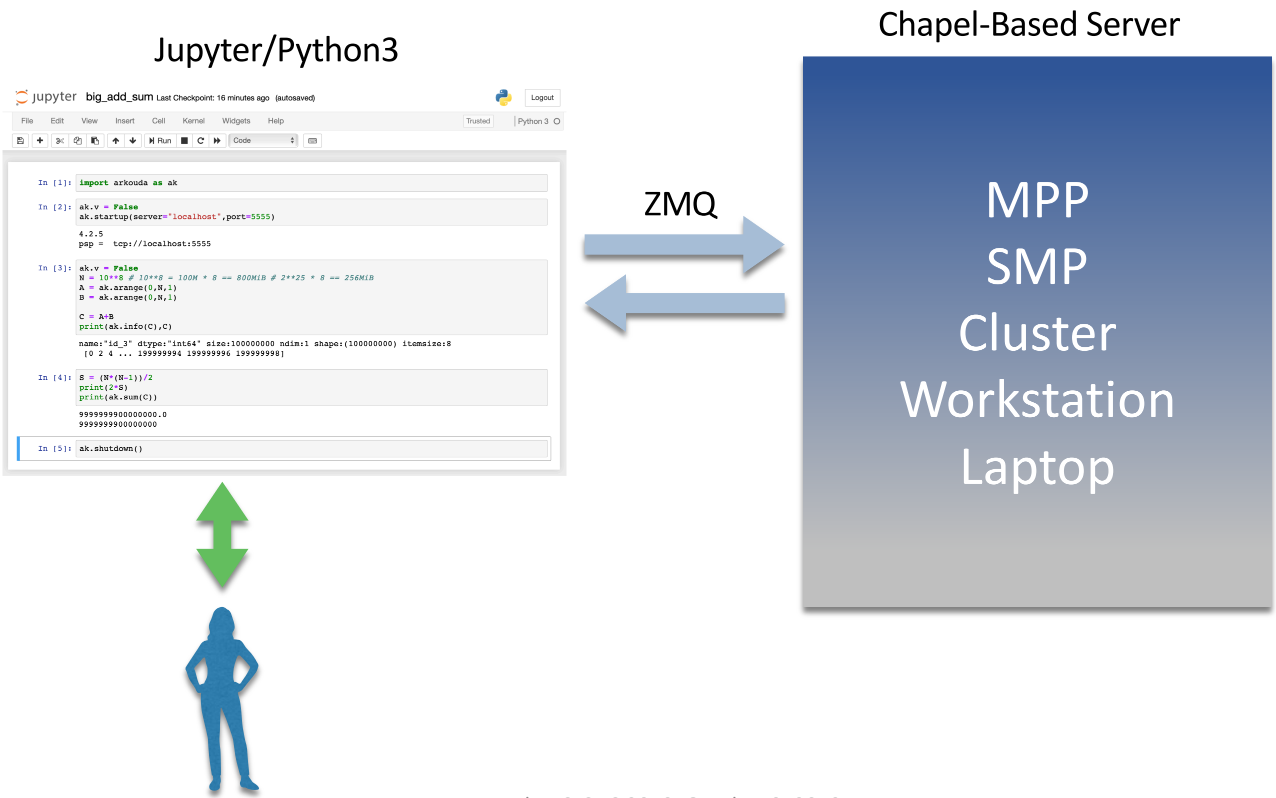
… and interoperate with optimized libraries.

Arkouda: an HPC shell for data science

- Jupyter/Python frontend (client)
- NumPy-like API
- Chapel backend (server)

based on material presented at CLSAC 2019, October 9, 2019

Arkouda Design



Arkouda Startup

1) In terminal:

```
> arkouda_server -nl 96

server listening on hostname:port
```

2) In Jupyter:

```
In [2]: import arkouda as ak
        ak.connect(hostname, port)

4.2.5
psp =  tcp://nid00104:5555
connected to tcp://nid00104:5555
```

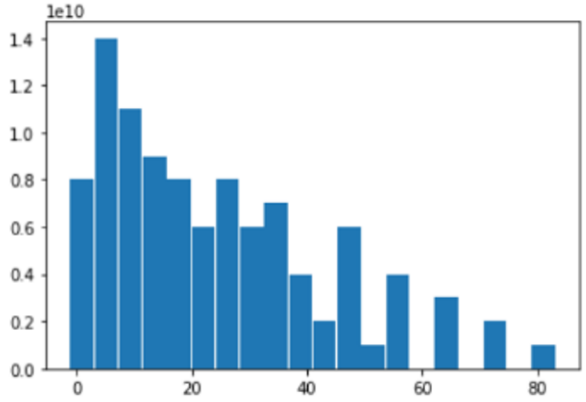
Data Exploration with Arkouda and NumPy

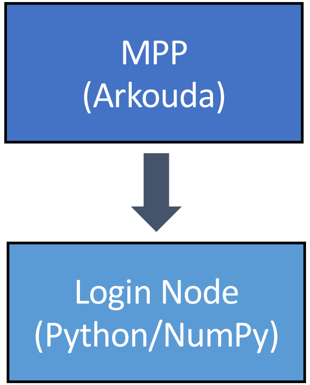
```
In [9]: A = ak.randint(0, 10, 10**11)
        B = ak.randint(0, 10, 10**11)
        C = A * B
        hist = ak.histogram(C, 20)
        Cmax = C.max()
        Cmin = C.min()
```

executed in 3.96s, finished 13:45:28 2019-09-12

```
In [10]: bins = np.linspace(Cmin, Cmax, 20)
         _ = plt.bar(bins, hist.to_ndarray(), width=(Cmax-Cmin)/20)
```

executed in 193ms, finished 13:45:28 2019-09-12

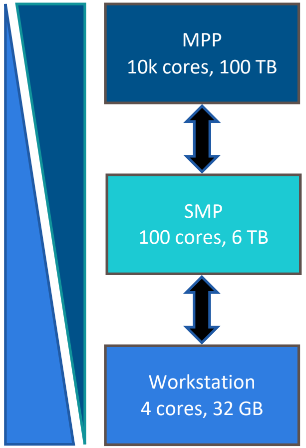




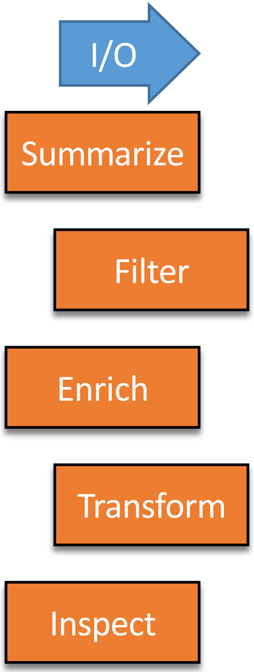
Arkouda Accomplishments

By taking this approach, these users were able to:

- interact with a running Chapel program from Python within Jupyter
- run the same back-end program on…
  - …a Mac laptop
  - …an Infiniband cluster
  - …an HPE Superdome X
  - …a Cray XC
- compute on TB-sized arrays in seconds
- with 1-2 person-months of effort



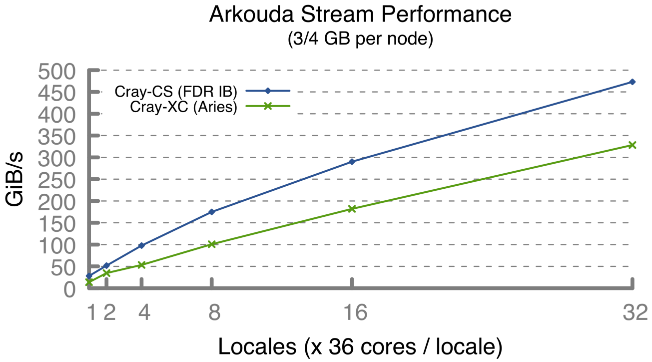
Hypothesis Testing on 50 Billion Records

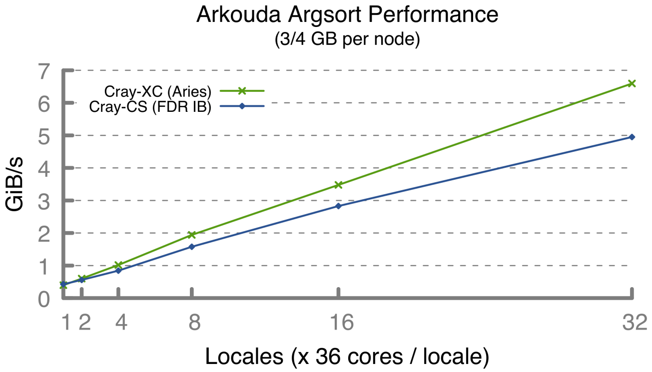


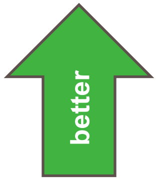
Operation	Example	Approximate Time (seconds)
Read from disk	A = ak.read_hdf()	30-60
Scalar Reduction	A.sum()	< 1
Histogram	ak.histogram(A)	< 1
Vector Ops	A + B, A == B, A & B	< 1
Logical Indexing	A[A == val]	1 - 10
Set Membership	ak.in1d(A, set)	1
Gather	B = Table[A]	30 - 300
Group by Key	G = ak.GroupBy(A)	60
Aggregate per Key	G.aggregate(B, ‘sum’)	15
Get Item	print(A[42])	< 1
Export to NumPy	A[:10**6].to_ndarray()	2

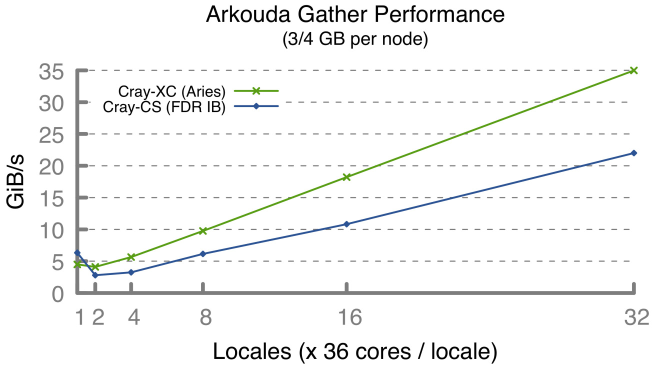
- A, B are 50 billion-element arrays
- Timings measured on real data
- Hardware: Cray XC40
  - 96 nodes
  - 3072 cores
  - 24 TB
  - Lustre filesystem

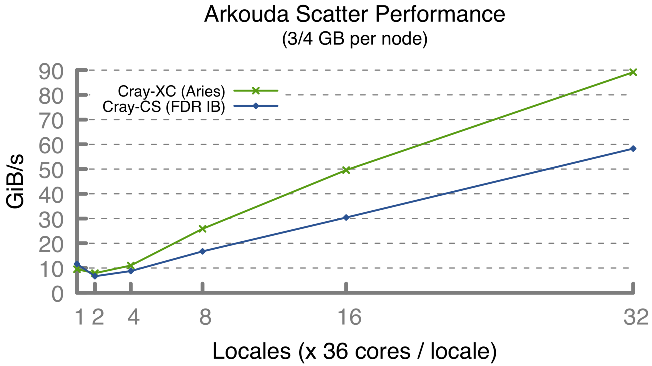
Arkouda Scaling: Aries vs. IBV (32 locales, 1152 locales)



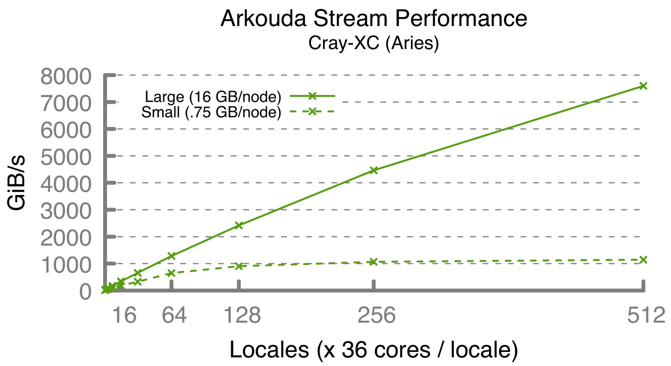


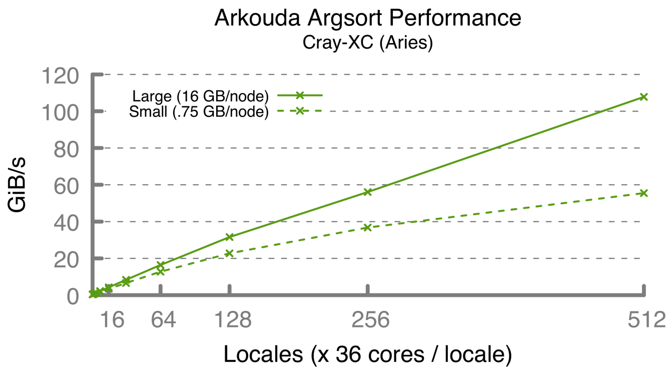


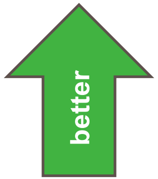


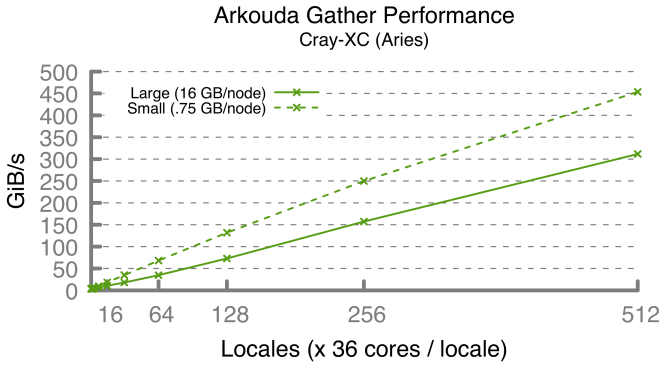


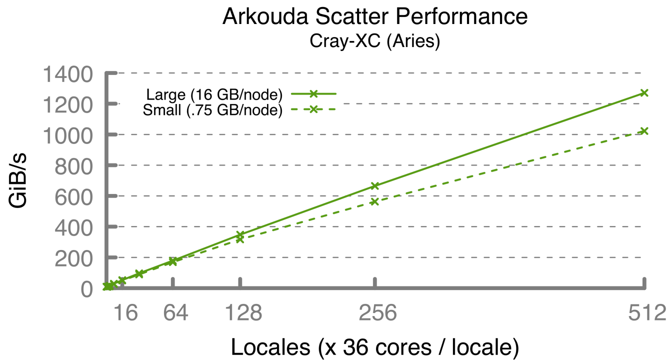
Arkouda Scaling: Aries at scale (512 locales, 18k cores)





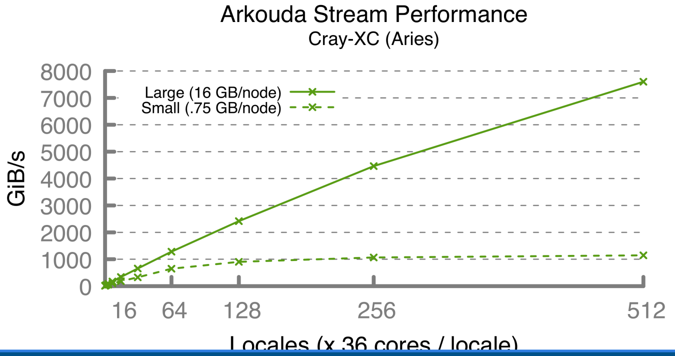


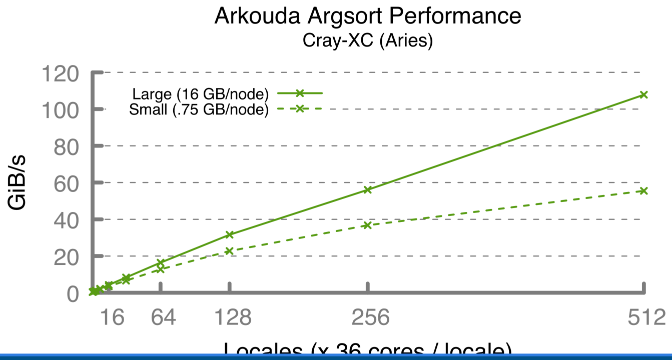


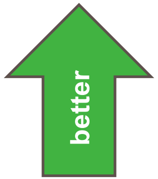


Arkouda Scaling: Aries at scale (512 locales, 18k cores)

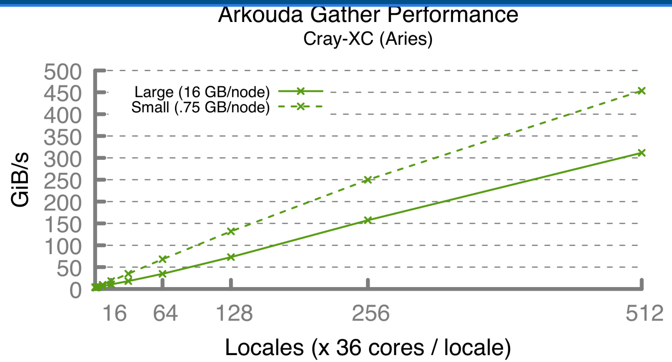
a Hewlett Packard Enterprise company

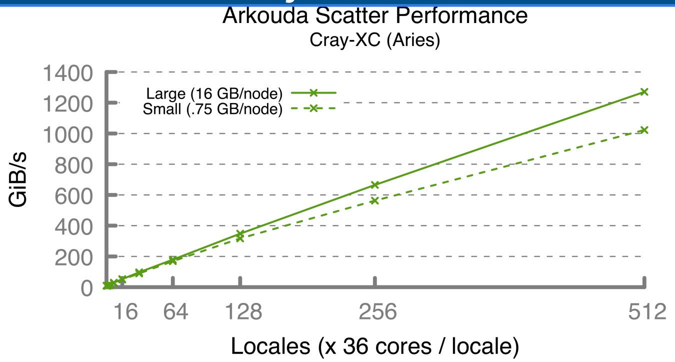






Sample result: Sorted 8TB of IPV4 addresses using 18k cores in just over a minute





Arkouda Design

- Why Chapel?
  - High-level language with C-comparable performance
  - Parallelism is a first-class citizen
  - Great distributed array support
  - Portable code: from laptop up to supercomputer

Arkouda Design

- Why Chapel?
  - High-level language with C-comparable performance
  - Parallelism is a first-class citizen
  - Great distributed array support
  - Portable code: from laptop up to supercomputer

Brad:

Also:

- Integrates with [distributed] numerical libraries (e.g., FFTW, FFTW-MPI)
- Close to “Pythonic” (for a statically typed language)
  - Provides a gateway for data scientists ready to go beyond Python

“Why not…”

“…Dask?”

- Didn’t want to be stuck in Python / wanted to run closer to the metal
- Found that it didn’t perform / scale well in their experience

Arkouda Status

- Now 11,000+ lines of Chapel code, developed in one year
  - “without Chapel, we could not have gotten this far this fast”

- Recently open-sourced
  - being developed on GitHub: https://github.com/mhmerrill/arkouda
  - available via ‘pip install’

- Being used on a daily / weekly basis on real data and problems
  - Features being added as requested by users

Current Arkouda Focus Areas

- Permit users to inject newly coded data filters into Arkouda as it's running
- Expand API
  - actual dataframes (currently informal collections of arrays)
  - sparse matrix computations
  - wrapping existing HPC libraries
- Improve performance / scalability
  - esp. on non-XC systems (e.g., IBV, Superdome)
- Outreach / Community development
  - e.g., Salishan, DOE, CUG, SciPy, PuPPy…

Arkouda Summary

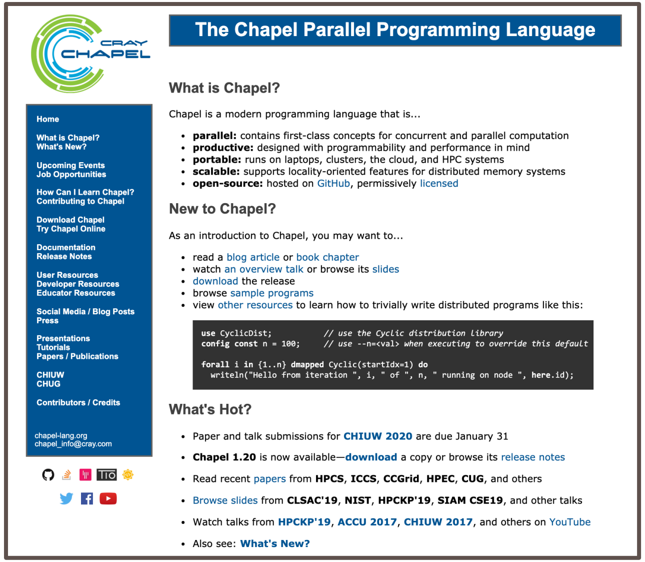
- A powerful tool and vision
  - "NumPy/Pandas on TB-scale arrays in seconds to minutes"
  - "a workbench for interactive HPC-scale data science"

- A great killer app for Chapel
  - **productivity:** decreased time-to-solution where time was of the essence
  - **scalability:** permits analyzing massive data sets
  - **performance:** supports interactive rates (seconds to minutes)
  - **portability:** across multiple system types and scales

For More Information

- **Arkouda GitHub:** https://github.com/mhmerrill/arkouda
- **Arkouda PyPi page:** https://pypi.org/project/arkouda/
- **Arkouda Gitter Channel:** https://gitter.im/ArkoudaProject/community
- **Bill Reus's CLSAC talk:** http://www.clsac.org/uploads/5/0/6/3/50633811/2019-reus-arkuda.pdf

- **Chapel website:** https://chapel-lang.org



SAFE HARBOR STATEMENT

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts.

These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.