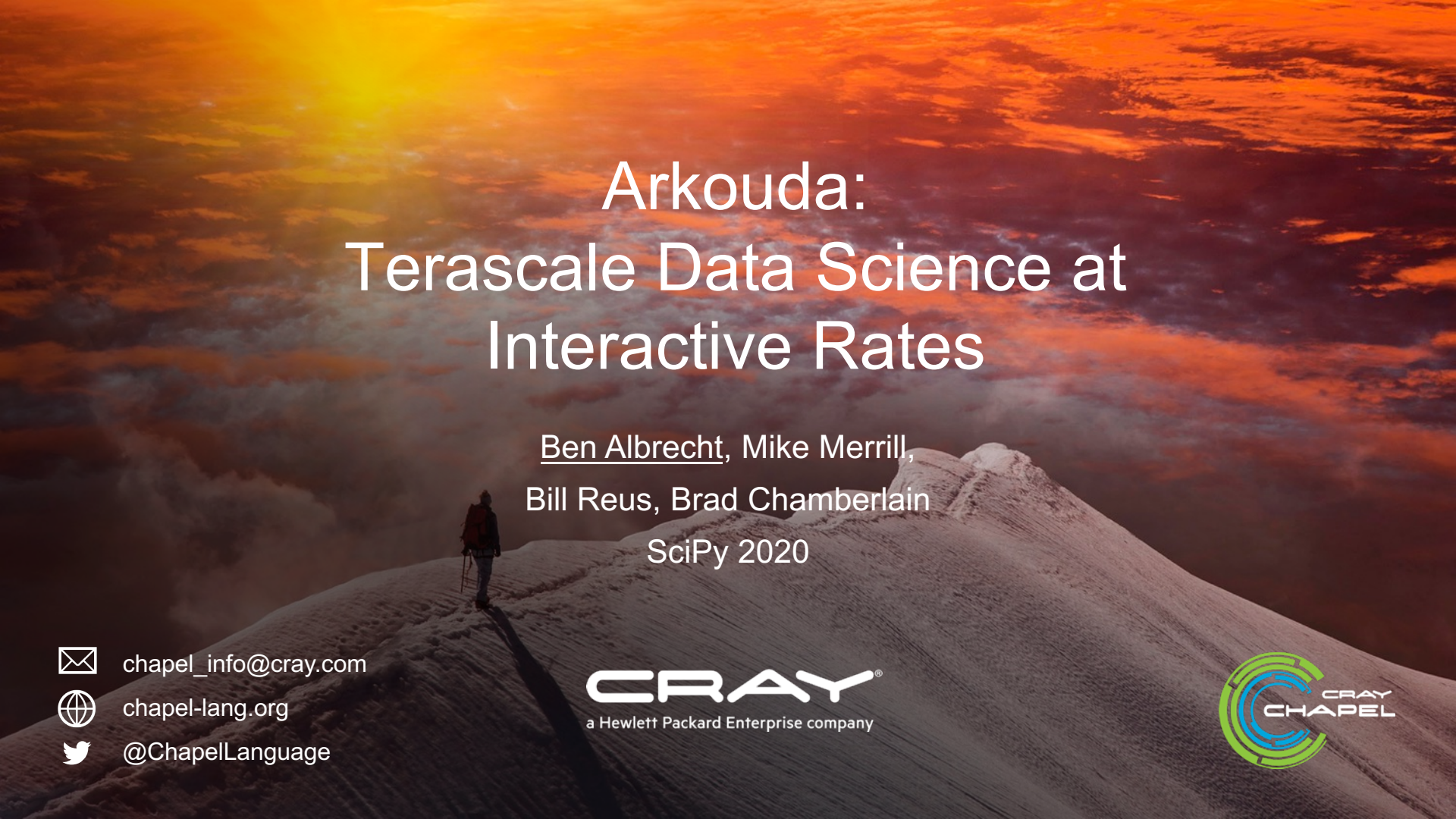


Arkouda: Terascale Data Science at Interactive Rates

Ben Albrecht, Mike Merrill,
Bill Reus, Brad Chamberlain

SciPy 2020

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

CRAY
a Hewlett Packard Enterprise company

CRAY CHAPEL

Motivation for Arkouda

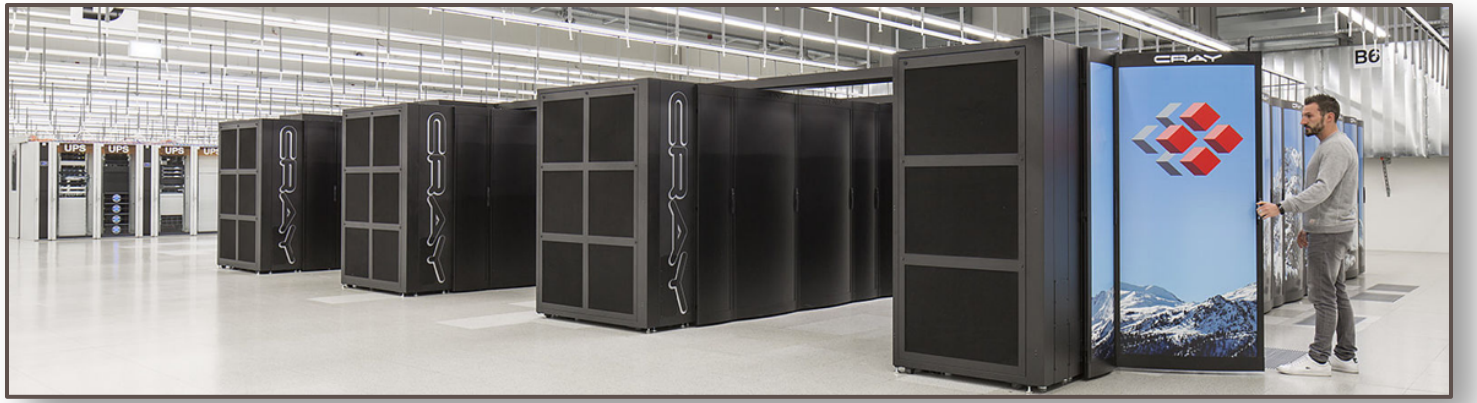
Motivation: Say you've got…

…a bunch of Python programmers

…HPC-scale problems to solve

…access to HPC systems

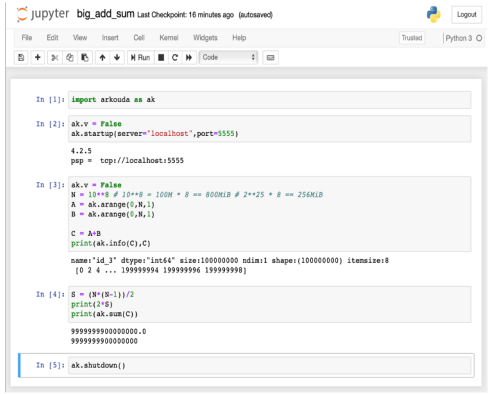
https://www.cscs.ch/computers/piz-daint/



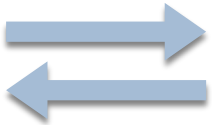
How will you leverage your Python programmers to get your work done?

Arkouda Design

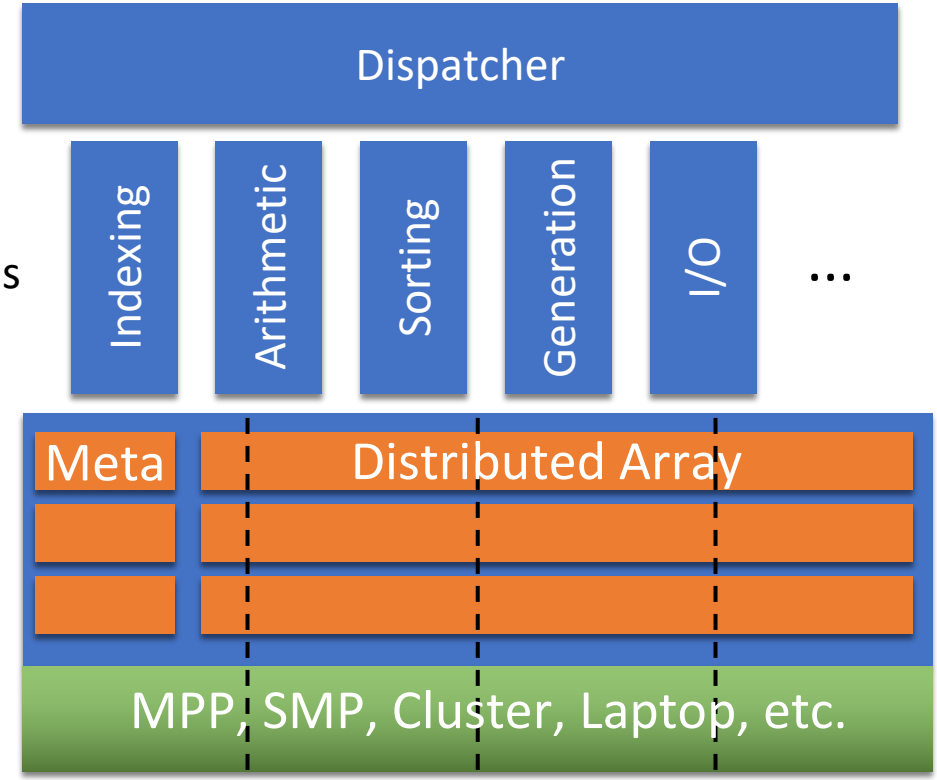
Python3 Client



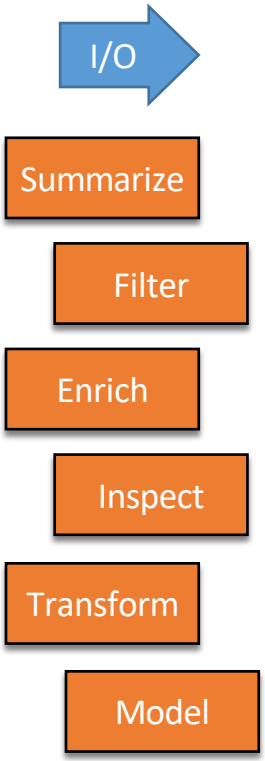
ZMQ Socket



Chapel Server



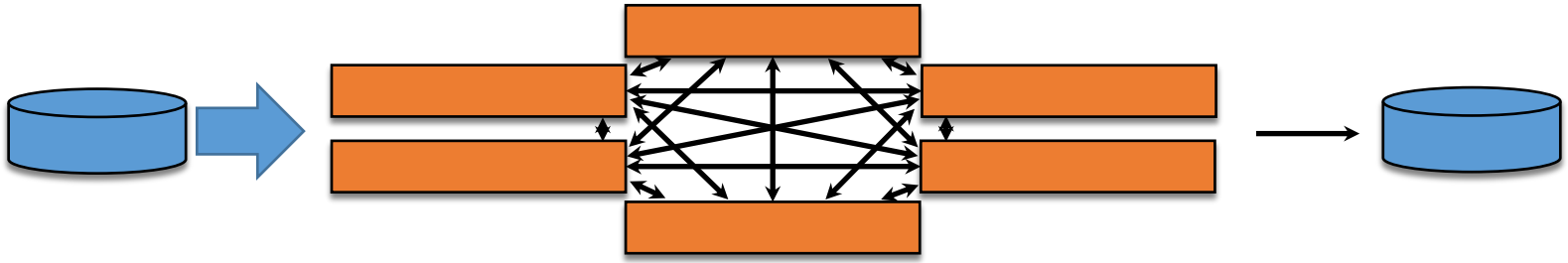
Data Science on 50 Billion Records



Operation	Example	Approx. 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[B == val]	1 - 10
Set Membership	ak.in1d(A, set)	1
Gather	B = Table[A]	4 - 120
Get Item	print(A[42])	< 1
Sort Indices by Value	I = ak.argsort(A)	15
Group by Key	G = ak.GroupBy(A)	30
Aggregate per Key	G.aggregate(B, 'sum')	10

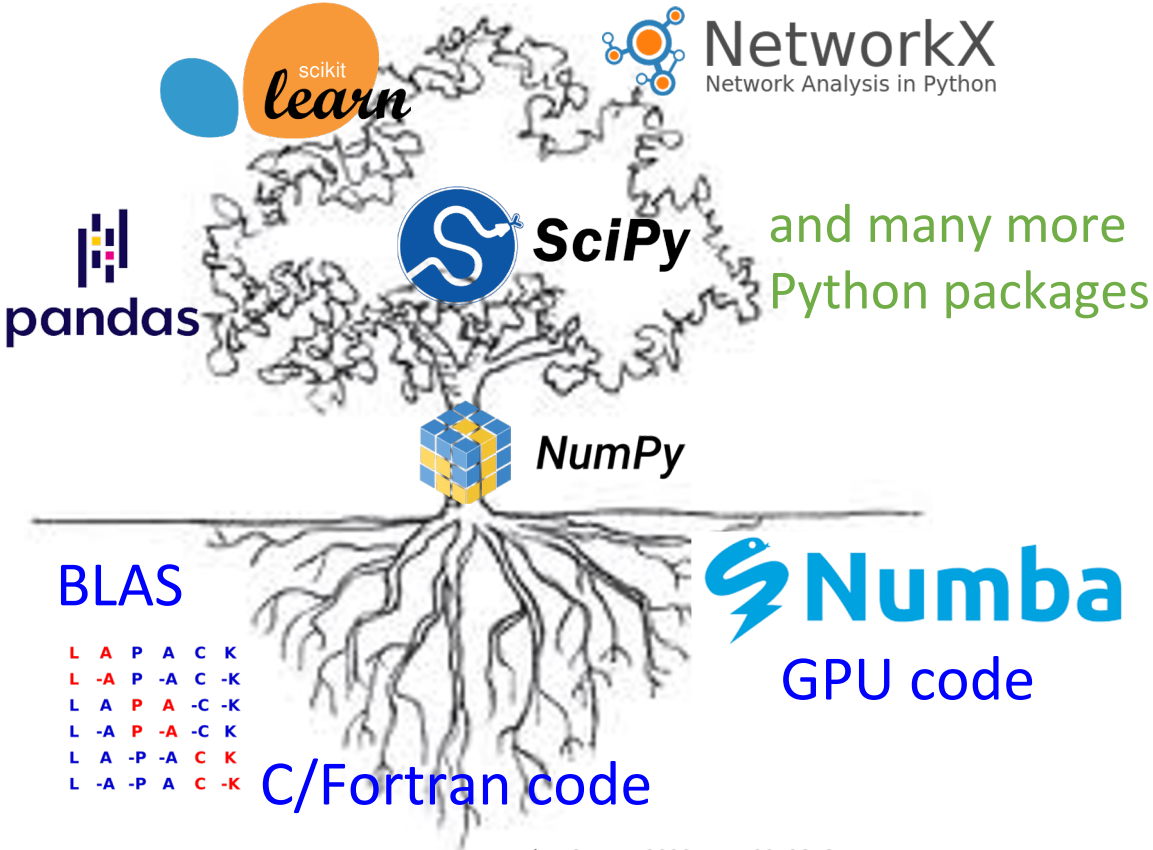
- A, B are 50 billion-element arrays of 32-bit values
- Timings measured on real data
- Hardware: Cray XC40
  - 96 nodes
  - 3072 cores
  - 24 TB
  - Lustre filesystem

Data Science Demands Interactivity

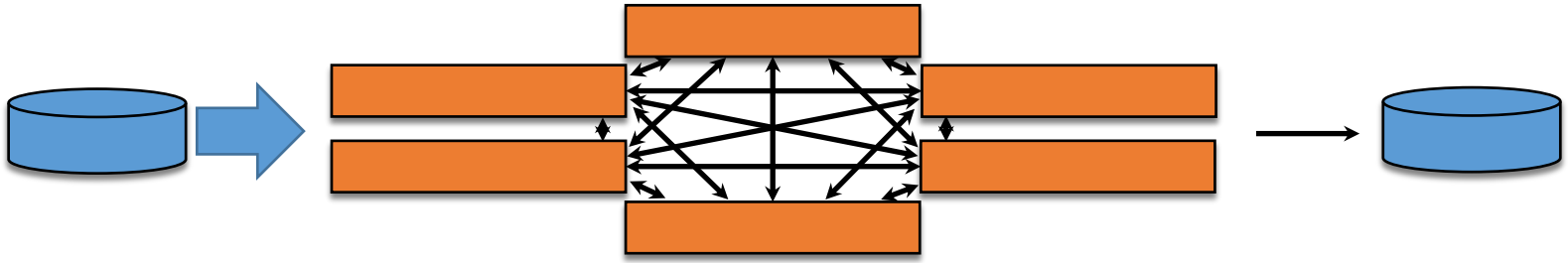


- Productivity with just enough performance
  - No compilation
  - No intermediate I/O
  - No writing boilerplate code
  - *Fast enough* to stay within thought loop
- Interactive Python on a large server satisfies these criteria for datasets up to 10-100 GB

Python Is Not Really Python



Data Science Demands Scaling



- Must use the whole dataset
  - Unbiased sampling of large datasets is difficult
  - Even unbiased sampling eliminates rare and high-order effects
  - Physics of most datasets are global, not local
- Datasets have outgrown (normal) computers
  - Server memory: ~ 1 TB
  - Many datasets > 10 TB

Arkouda

Load Terabytes of data…

… into a familiar, interactive UI …

… where standard data science operations …

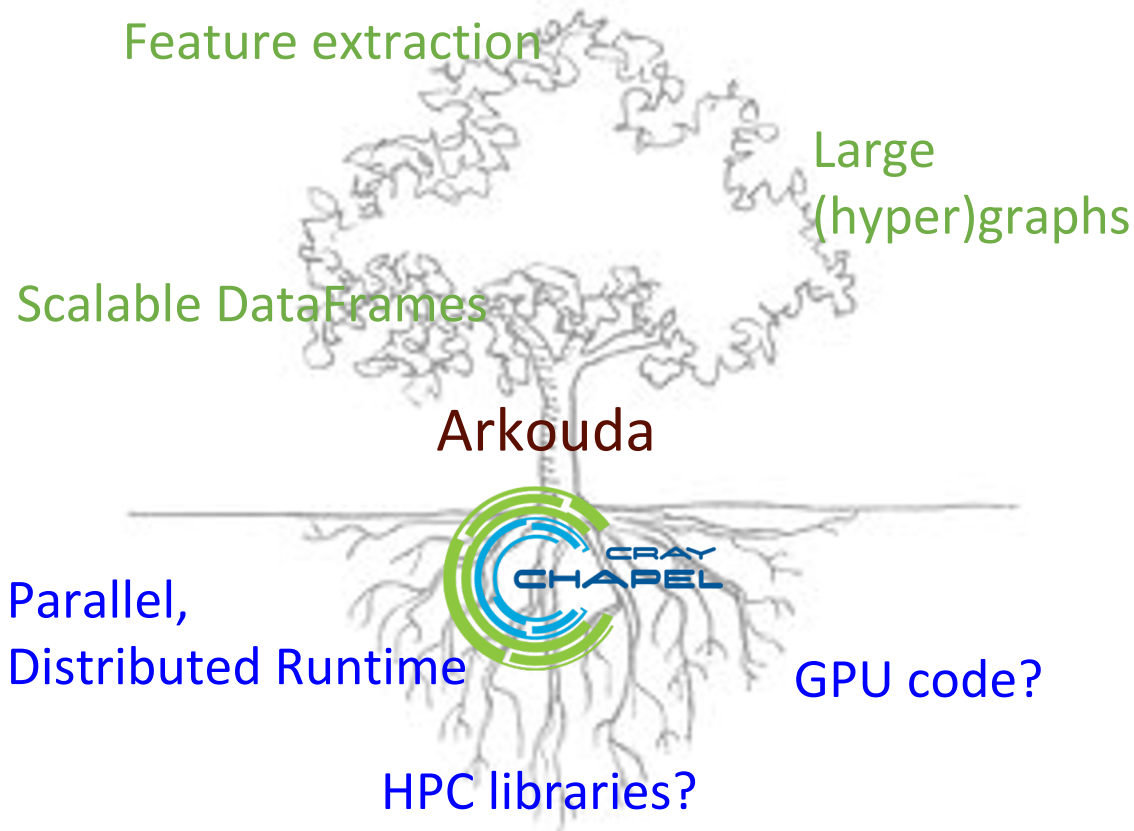
… execute within the human thought loop …

… and interoperate with optimized libraries.

Arkouda: an HPC shell for data science

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

Arkouda: NumPy for HPC



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

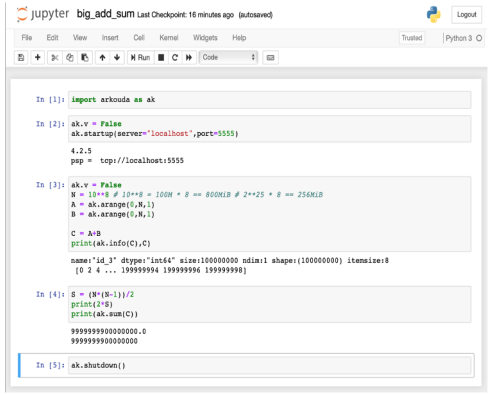
Why Chapel?

- From the lead Arkouda developers:
  - High-level language with C-comparable performance
  - Parallelism is a first-class citizen
  - Portable code: from laptop up to supercomputer
  - Integrates with (distributed) numerical libraries (e.g., FFTW, FFTW-MPI)
  - Close to “Pythonic” (for a statically typed language)
    - Lowers the barrier for users to modify the backend implementation

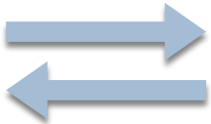
```
var D = {1..1000, 1..1000} dmapped Block(...),
    A: [D] real;

forall (i,j) in D do
  A[i,j] = i + (j - 0.5)/1000;
```

Arkouda Design

Python3 Client



ZMQ Socket



Chapel Server

Dispatcher

Code Modules

Indexing | Arithmetic | Sorting | Generation | I/O | ...

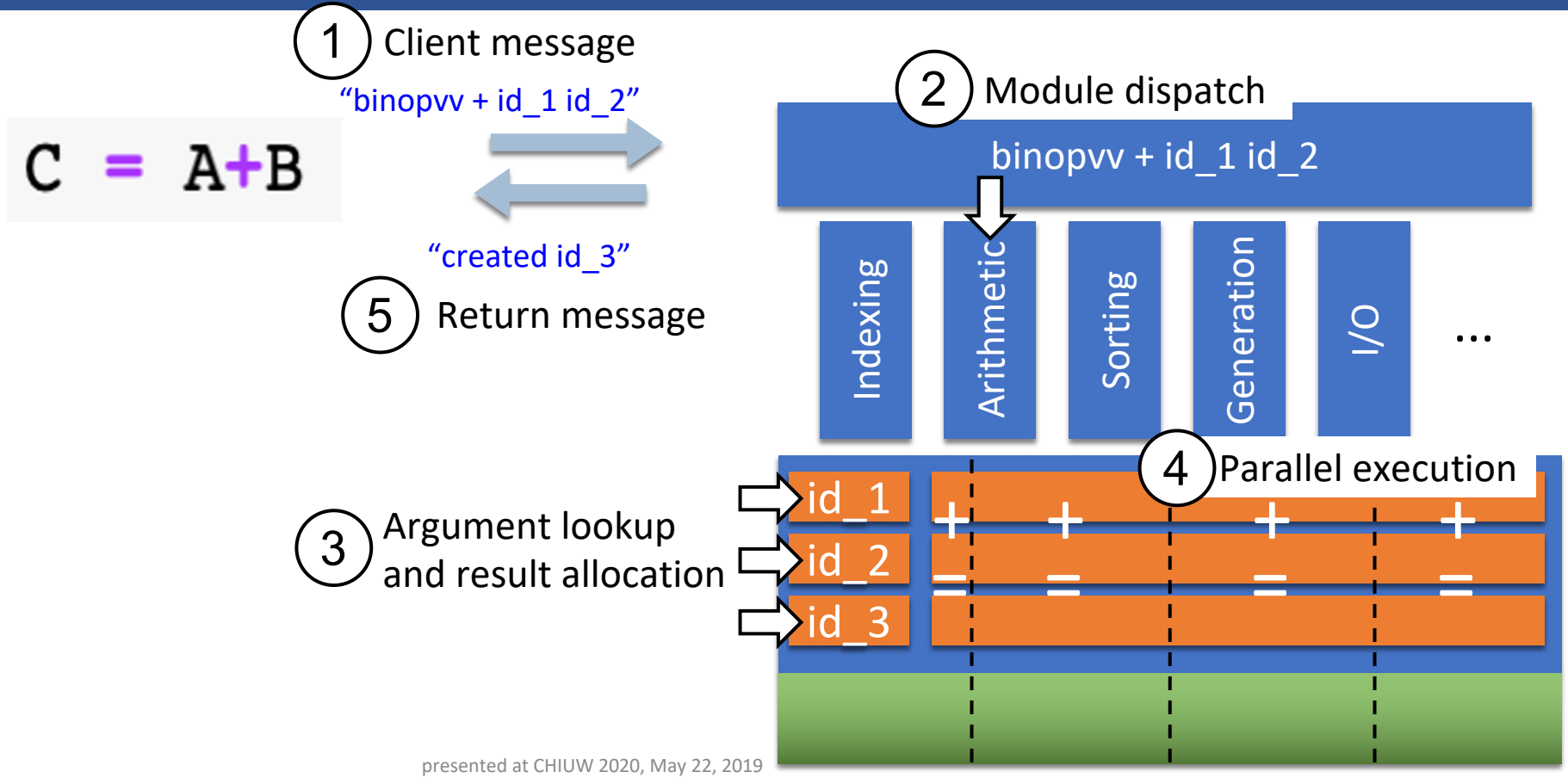
Distributed Object Store

Meta | Distributed Array

Platform

MPP, SMP, Cluster, Laptop, etc.

A Chapel Interpreter



Using Arkouda: Startup

1) Initialize server in terminal

```
> arkouda_server -nl 96


server listening on hostname:port
```

2) Connect to server in Jupyter

```
import arkouda as ak
ak.connect(hostname, port)
```

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

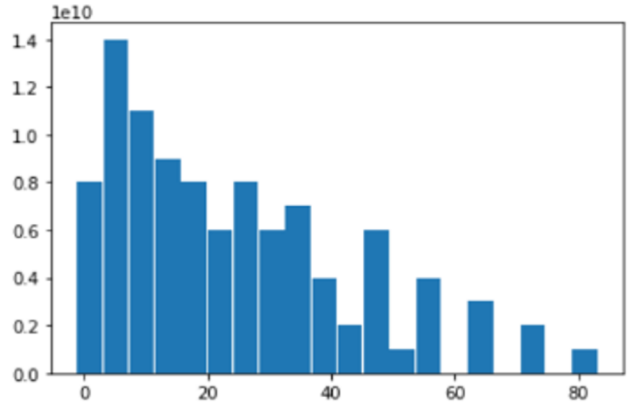
Using Arkouda: Workflow

```
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



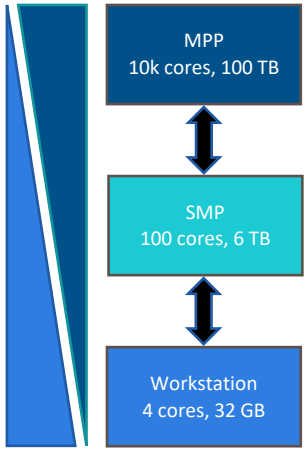
MPP
(Arkouda)

Login Node
(Python/NumPy)

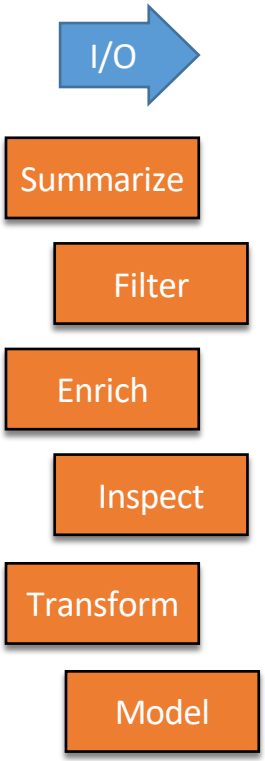
Arkouda Accomplishments

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



Data Science on 50 Billion Records



Operation	Example	Approx. 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[B == val]	1 - 10
Set Membership	ak.in1d(A, set)	1
Gather	B = Table[A]	4 - 120
Get Item	print(A[42])	< 1
Sort Indices by Value	I = ak.argsort(A)	15
Group by Key	G = ak.GroupBy(A)	30
Aggregate per Key	G.aggregate(B, 'sum')	10

- A, B are 50 billion-element arrays of 32-bit values
- Timings measured on real data
- Hardware: Cray XC40
  - 96 nodes
  - 3072 cores
  - 24 TB
  - Lustre filesystem

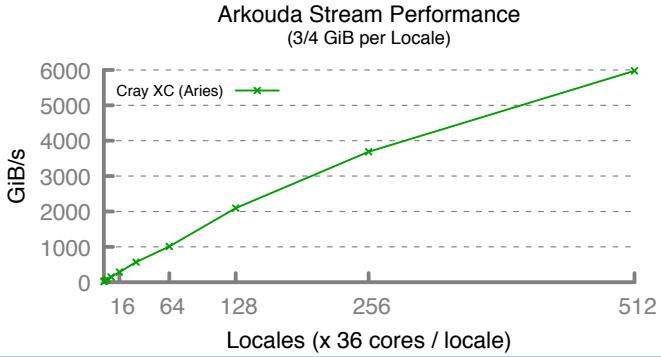
NumPy vs. Arkouda Performance

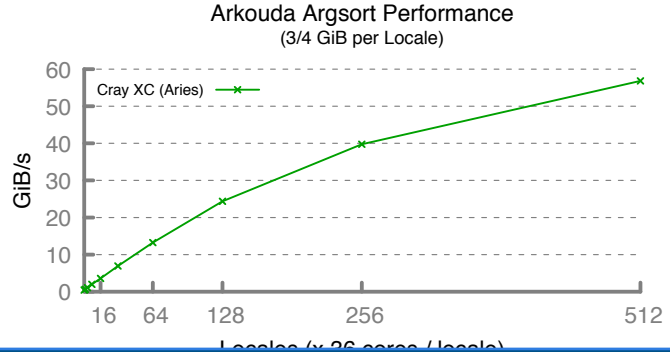
Performance normalized to NumPy

Cray XC (Aries network)
36-core (72 HT), 128 GB RAM
dual 18-core “Broadwell”, 2.1 GHz

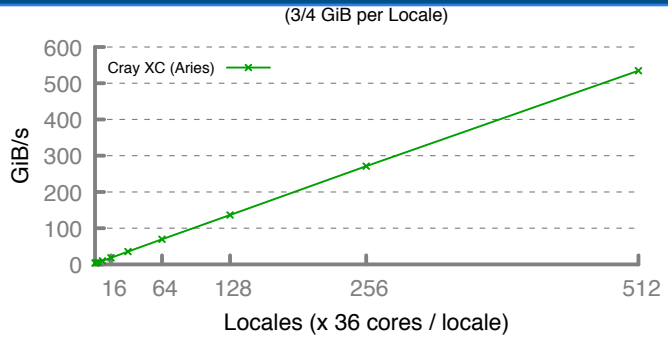
benchmark	NumPy	Arkouda (serial)	Arkouda (1-node/36-core)	Arkouda (512-node)
argsort	1.0	2.0	16.7	1837.3
coargsort	1.0	2.3	16.7	984.7
gather	1.0	0.4	11.7	469.1
reduce	1.0	1.2	12.0	4412.4
scan	1.0	0.8	3.2	266.6
scatter	1.0	1.0	11.8	781.8
stream	1.0	0.7	6.2	1590.4

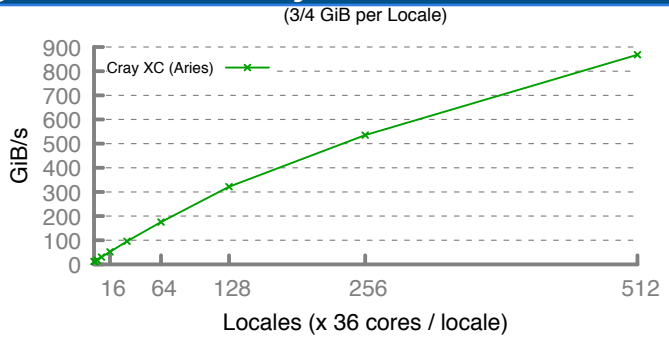
Arkouda Scaling: Aries at scale (512 locales, 18k cores)



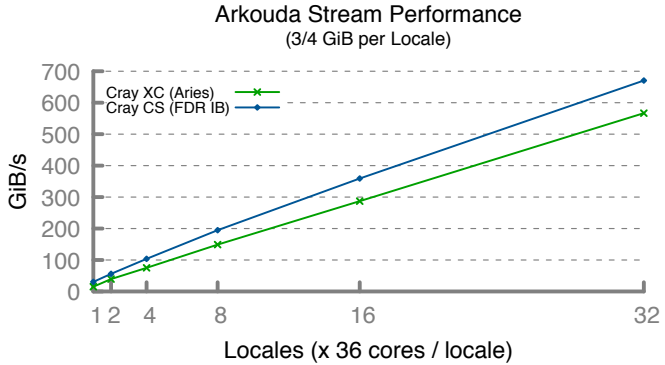


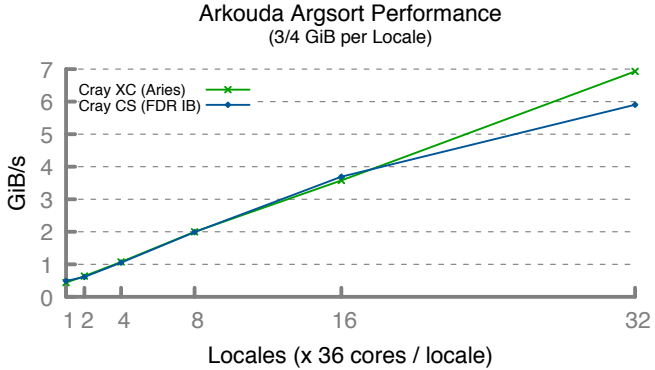
Sample result: Sorted 8TB of IPV4 addresses using 18k cores in just over a minute

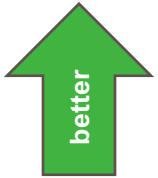


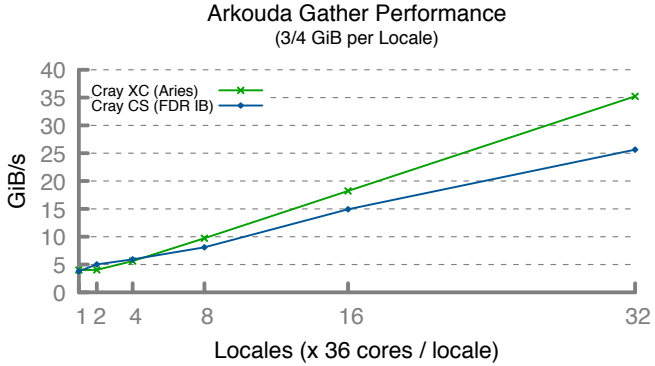


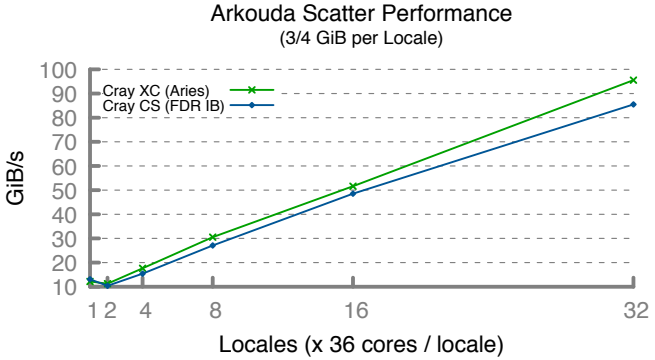
Arkouda Scaling: Aries vs. IBV (32 locales, 1152 cores)











What about Dask?

- Arkouda performs significantly faster in our experience*
  - Comparison involves reading from HDF5 file and doing a scalar reduction
  - *Looking to work with Dask experts to ensure a fair comparison
- Dask configuration on HPC systems is not trivial
  - Getting Arkouda performance out the box is easy
- Dask is not designed for shuffle-based workflows (sorting, grouping, etc.)
  - Arkouda targets these workflows and scales without lazy evaluation
- Dask is much more established and featureful
  - Better integration with Pandas and more supported data formats
  - Excellent cluster management tooling

Arkouda Status

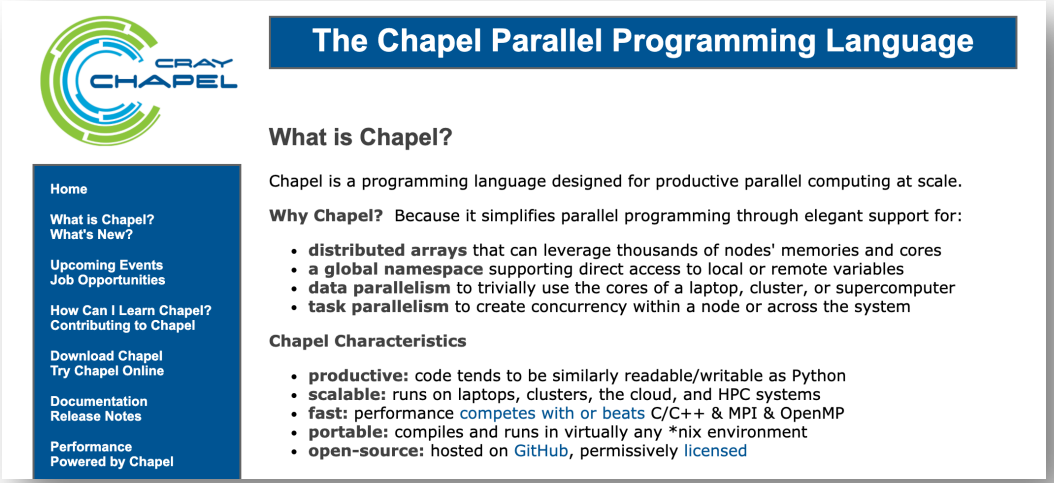
- Now 12,000+ lines of Chapel code, developed in one year
  - “without Chapel, we could not have gotten this far this fast”
- Open source:
  - Being developed on GitHub
  - Available as a PyPI package via ‘pip install arkouda’
- Being used on a daily / weekly basis on real data and problems
  - Features being added as requested by users

Arkouda Summary & Next Steps

- Arkouda is a powerful tool and vision
  - NumPy/Pandas on TB-scale arrays in seconds to minutes
  - A workbench for interactive HPC-scale data science
- Arkouda takes a unique approach to interactivity in HPC
  - Starts with performance and builds towards interactivity
- Next steps involve expanding API and supporting multi-user sessions
  - Actual dataframes (currently informal collections of arrays)
  - Sparse linear algebra (GraphBLAS)
  - Wrapping existing HPC libraries
  - Data sharing and access control

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 CHIUW talk:** https://chapel-lang.org/CHIUW2020.html#keynote
- **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.