FuncX: A Function Serving Platform for HPC

Ryan Chard

28 Jan 2019

globus labs

Outline

- Motivation
- FuncX: FaaS for HPC
- Implementation status
- Preliminary applications
    - Machine learning inference
    - Automating analysis
    - Metadata extraction







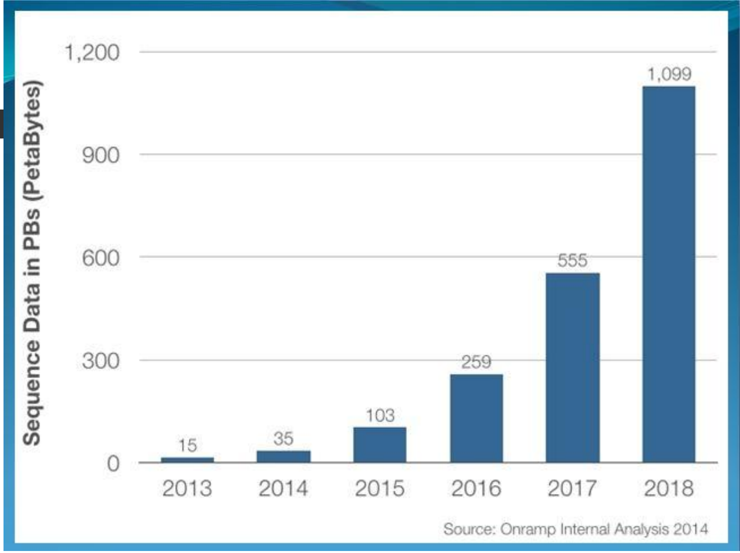


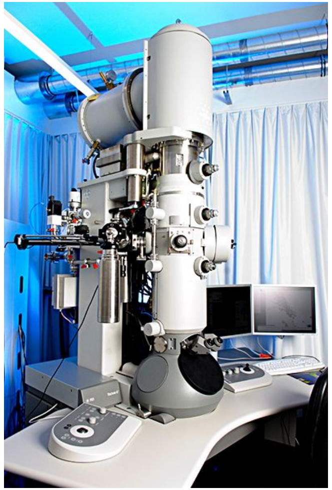

Next gen data

Data volumes and velocities are exploding, overwhelming local resource capabilities

Scientific results from almost all domains are increasingly computationally dependent

Instrument improvements mean a flood of new problems and users that could benefit from HPC





HPC Mismatch

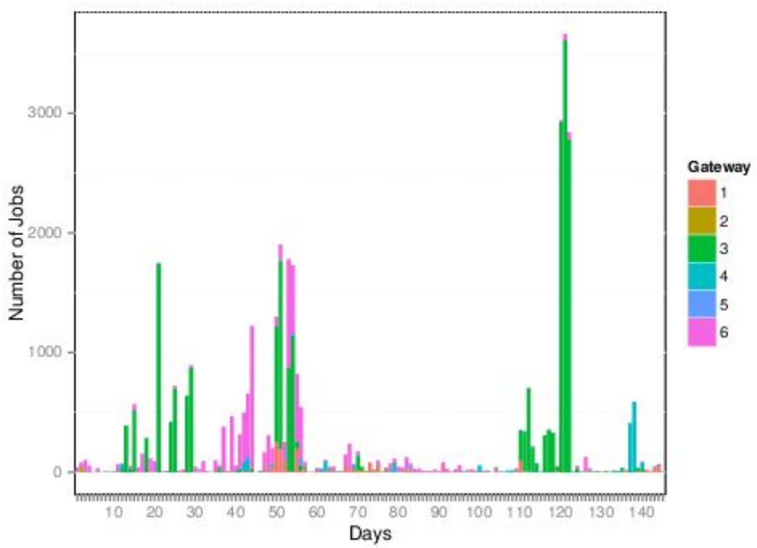
HPC often designed for extreme-scale workloads

Analysis requirements are often bursty

- SEMs don’t run 24/7
- Beamtime is rare
- Samples must be swapped
- Funding comes and goes

New paradigms are increasingly interactive

Growing need to accommodate analysis at scale, on-demand



Barriers to HPC

HPC environments aren’t user friendly

- Variety of platforms (architectures and accelerators)
- Steep learning curves (package management)
- Different interfaces (Slurm/PBS/Cobalt)
- Difficult to acquire resources (in a timely manner)
- Numerous modalities and frameworks for scaling
- Can’t hold resources without work

Small, on-demand tasks are not necessarily a priority

However, thousands of small tasks are a big problem

Serverless computing

Serverless computing is revolutionizing business IT

Function as a Service (FaaS)

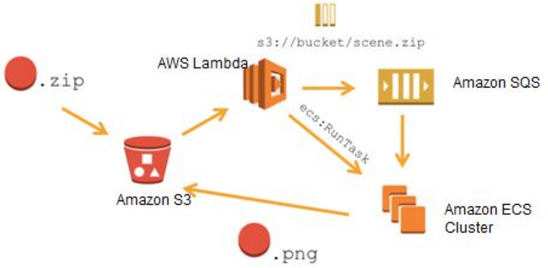
- Pick a runtime (python/JS/R etc.)
- Write function code
- Run at any scale

Low latency, on-demand

Can compose functions to solve complex problems

Google Cloud Functions

APACHE OpenWhisk™



Function serving for Science

1. Remove barriers, simplify usage, and federate access to HPC resources

2. Support a new generation of users and applications

3. New opportunities for optimization

1. Remove barriers

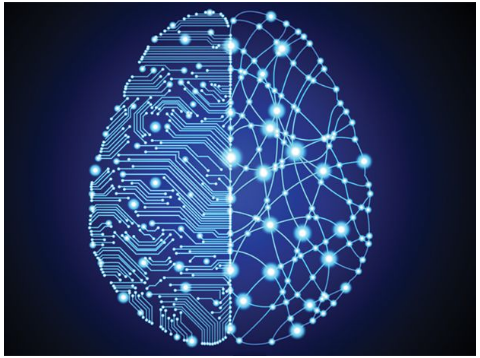
FaaS can make HPC accessible

- Abstract compute, only expose function code
- Containerized ***runtimes*** encapsulate dependencies
- Libraries of functions promote sharing and reuse
- FaaS service can provide secure, programmatic access

2. Support new applications

FaaS enables new applications for HPC

- Short duration and/or low-latency tasks
  - Real-time usage
  - Guide experiments
  - Interactive computing
  - ML inference
  - Stream processing



3. Optimization opportunities

FaaS can improve usage and utilization

- Locality aware function placement
  - Send queries to datasets
  - Allow multiple functions to share caches/datasets
- Share runtimes for rapid serving
- Use backfill queues to increase resource utilization
- Use HPC investments for new problems

FuncX: A FaaS platform for HPC

FuncX: FaaS for HPC

Enable secure, isolated, on-demand function serving on HPC resources for the masses

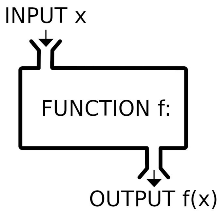
Abstract underlying infrastructure

Establish a library of functions, encouraging reuse and reproducibility

Functions and Runtimes

Functions

- Small executable codes
- Short duration tasks
- Stateless
- Invoked on-demand
- Accept input (JSON/binary), return output



Runtimes

- Container of libraries and dependencies
- Isolates function execution
- Serve multiple functions with shared dependencies
- “Warm” runtimes for rapid serving

Executing functions with Parsl

Parallel Scripting Library for Python

Annotate Python scripts with Parsl directives

Dynamically intercepts function invocation and creates DAG with data dependencies

Manages the execution of the script on clusters, clouds, grids, and other resources

Supports secure authentication (2FA)

```
@python_app
def hello ():
    return 'Hello World!'

print(hello().result())
```

```
Hello World!
```

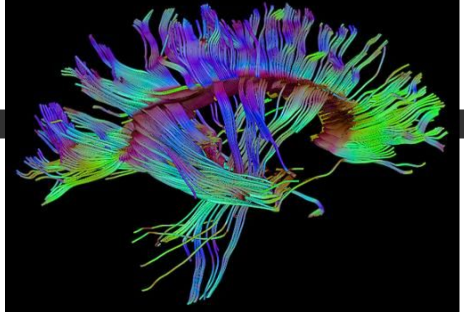
```
@bash_app
def echo_hello(stdout='echo-hello.stdout'):
    return 'echo "Hello World!"'

echo_hello().result()

with open('echo-hello.stdout', 'r') as f:
     print(f.read())
```

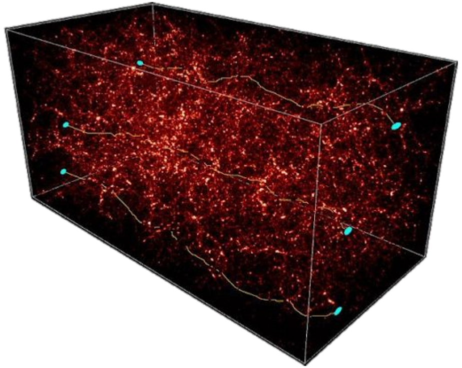
```
Hello World!
```

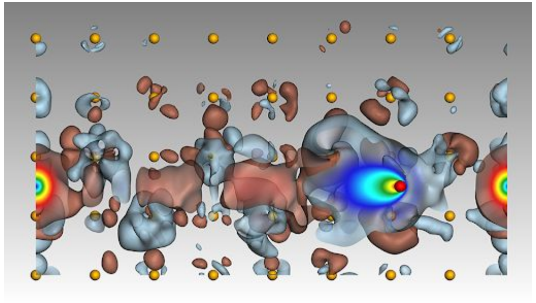
Parsl



Unique executors to meet application requirements

- High throughput (HTEX)
- Extreme scale (EXEX)
- Low latency (LLEX)





Abstracts resource integration

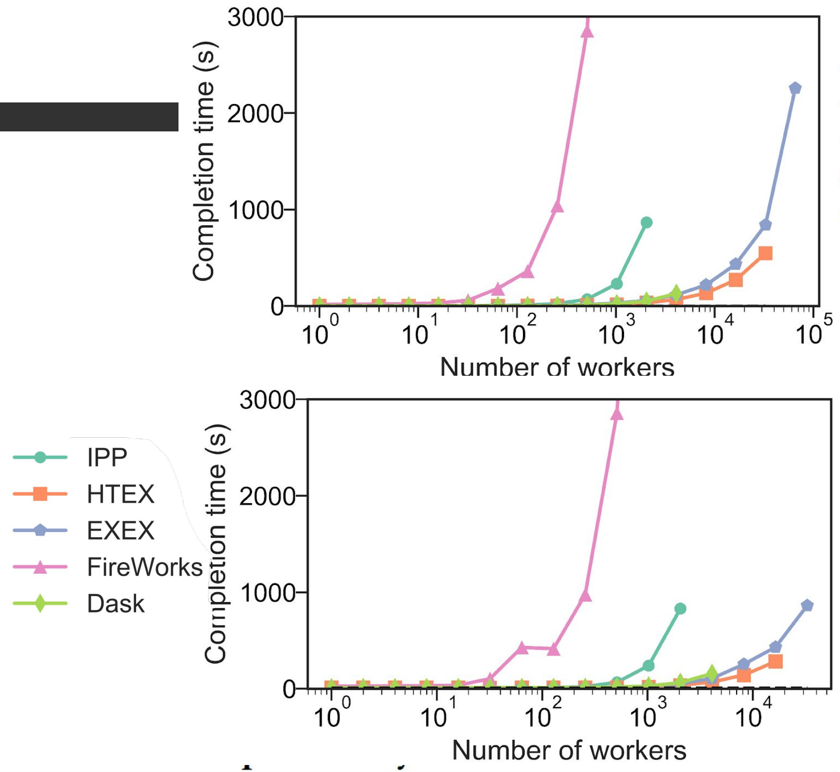
```
config = Config(
    executors=[
        HighThroughputExecutor(
            label="stampede2_htex",
            address=address_by_hostname(),
            provider=SlurmProvider(
                channel=LocalChannel(),
                nodes_per_block=128,
                init_blocks=1,
                partition="skx-normal",
                walltime="12:00:00"
            )
        )
    ]
)
```

Parsl scaling

Weak scaling: 10 tasks per worker. Task duration from 0 to 1s.

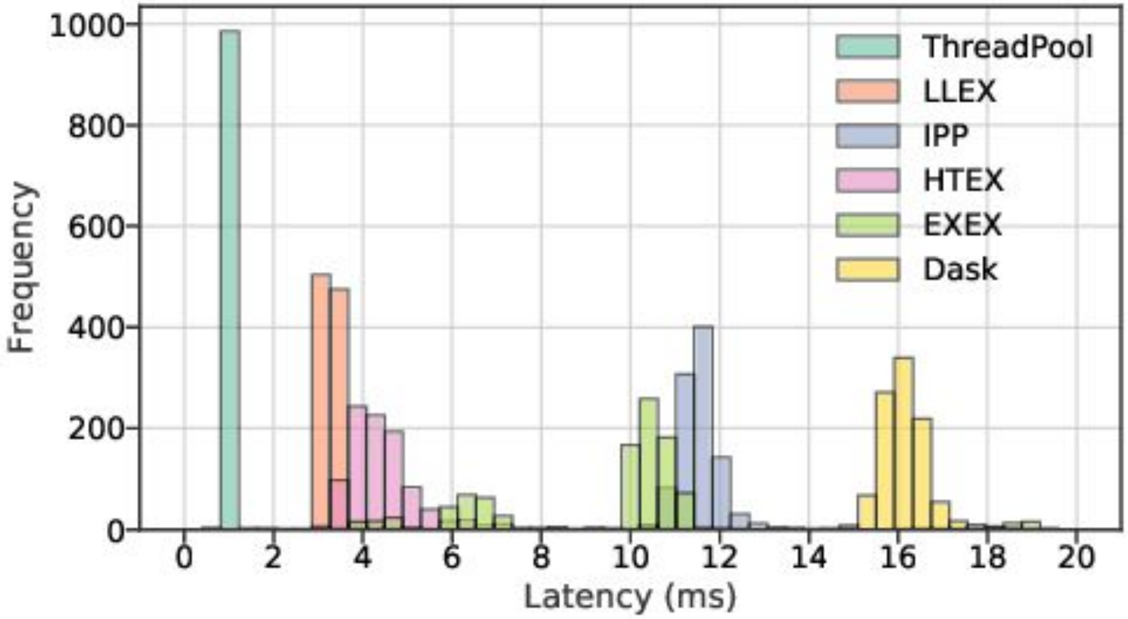
HTEX and EXEX outperform other Python-based approaches

HTEX and EXEX scale to 1K* and 8K* nodes, respectively, with >1K tasks/s



Framework	Maximum # of workers†	Maximum # of nodes†	Maximum tasks/second‡
Parsl-IPP	2048	64	330
Parsl-HTEX	32 768	1024*	1181
Parsl-EXEX	262 144	8192*	1176
FireWorks	1024	32	4
Dask distributed	4096	128	2617

Parsl low latency executor



LLEX achieves low (3.47ms) and consistent latency

HTEX (6.87ms) and EXEX (9.83) are less consistent

All executors are faster than IPP (11.72ms) and Dask (16.19ms)

Data management: Globus

Moving both data and runtimes

Auth and logging

- Identity management
- Authentication

Data staging

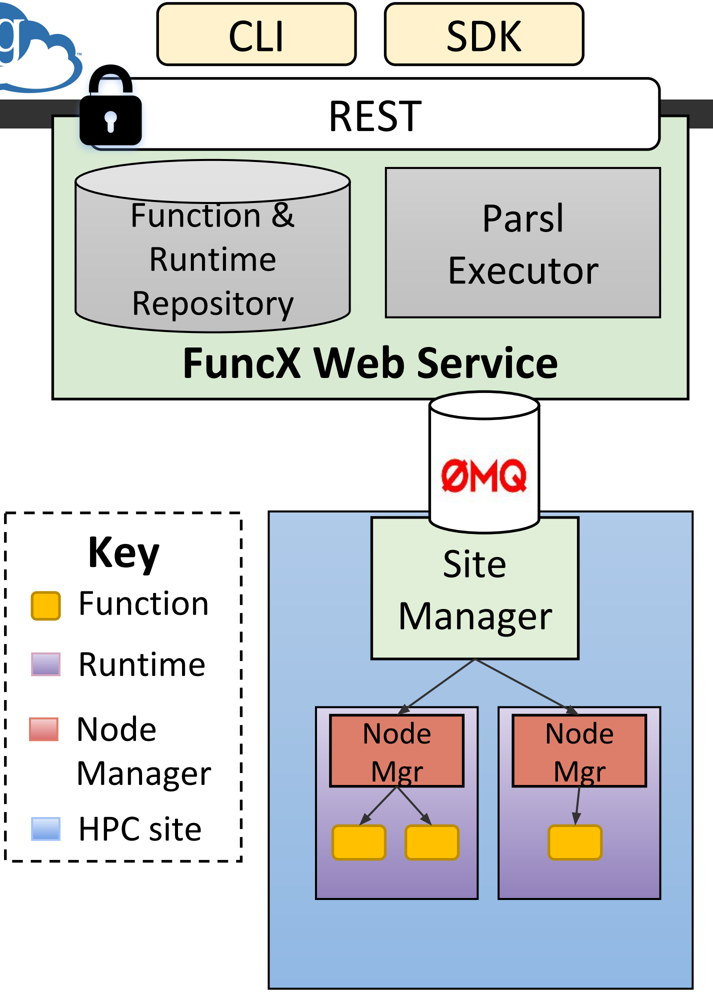
- Transfer to endpoints
- Stage to runtimes

FuncX Prototype

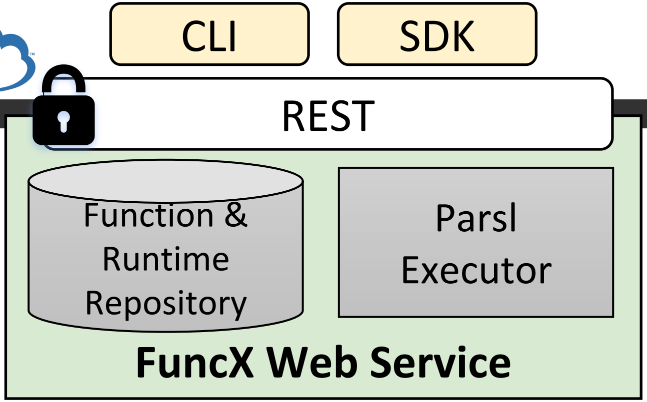
FuncX prototype

- Web service
  - Interact with FuncX
- Site manager
  - Deploy at HPC site
  - Manage runtimes at site
- Node manager
  - Manage functions within a runtime

Implementation status: ongoing



Web service



REST API to create, invoke, delete functions

Dynamically create runtime containers

- Record requirements to determine reuse
- Dockerize -> ECR -> singularity -> site managers

Relies on Globus Auth for identity and access management (IAM)

Parsl interchanges route jobs to active site managers

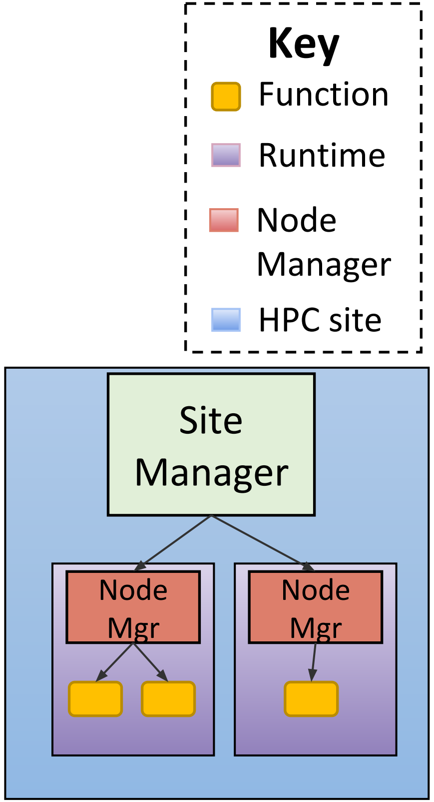
Log usage for user accountability

- Understand apps we are running

Site manager

Runtime management

- Maintain local runtime repository
- Deploy and manage runtimes
- Pass serialized functions and inputs into runtimes for execution
- Spin down “used” runtimes when idle
- Restrict access for “used” runtimes



Currently assume one site manager

- Single-, not multi-tenant
- Functions run as my own user on ALCF’s Cooley

Node manager

*** Currently under development ***

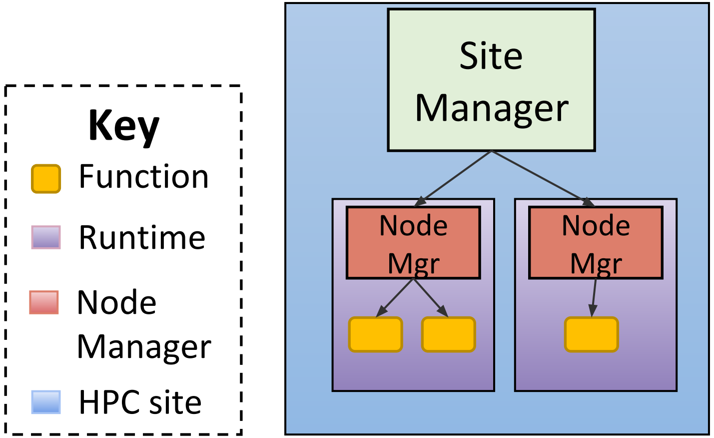
Deployed within a runtime to manage/execute functions

Responsible for creating sandboxes and instantiating functions

- UIDs and directories or linux containers...

Stage data to functions

Manage local cache

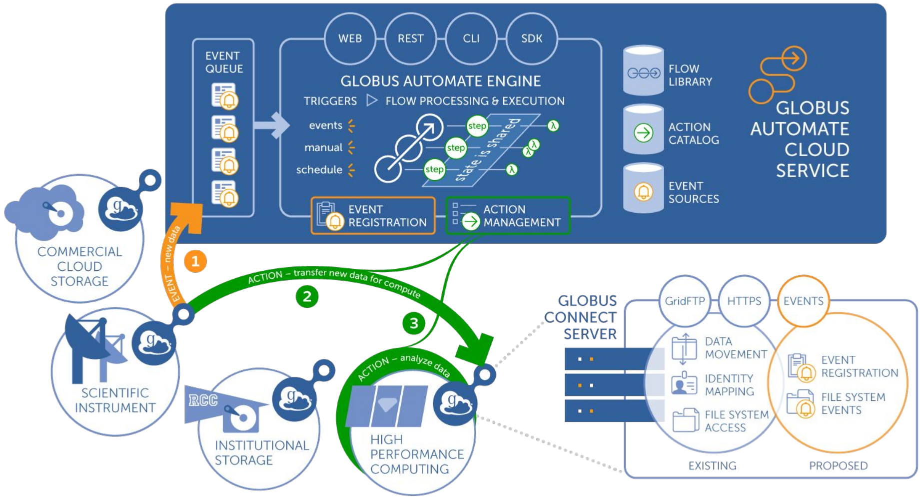


Preliminary work

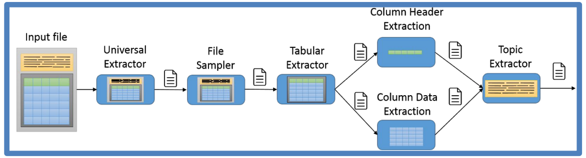
DLHub

Data and Learning Hub for Science

Globus Automate



Extract



DLHub

Collect ⟷ Serve ⟷ Train

- <u>Collect, publish, categorize</u> models from many disciplines
- <u>Serve</u> models via API to foster sharing, consumption, and access to data, training sets, and models
- <u>Simplify training</u> of models (using HPC and cloud)
- <u>Enable new science</u> through reuse and synthesis of existing models

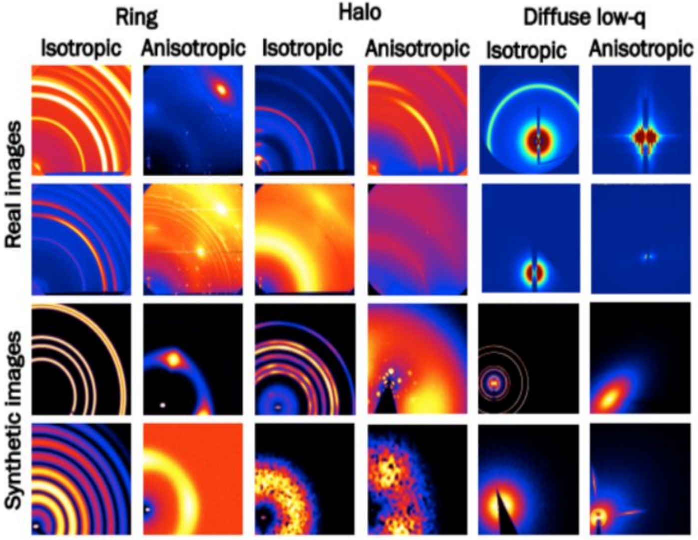
Funding: 2018 Argonne Adv. Computing LDRD

Argonne NATIONAL LABORATORY ARGONNE LEADERSHIP COMPUTING FACILITY PETREL parsl

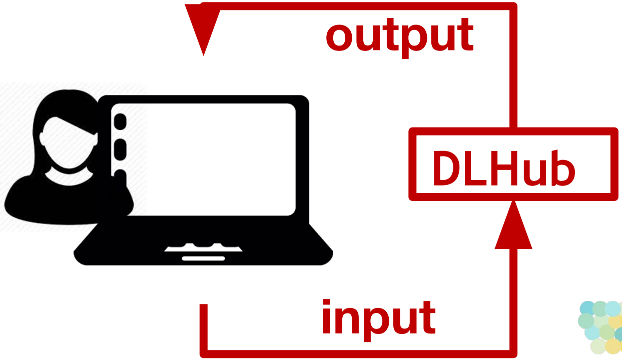
DLHub

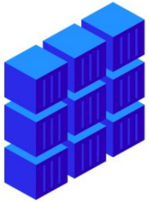
Robust and Scalable Deep Learning for X-ray Synchrotron Image Analysis

Nicole Meister[1*], Ziqiao Guan[2*], Jinzhen Wang[3], Ronald Lashley[4], Jiliang Liu[5], Julien Lhermitte[5], Kevin Yager[5], Hong Qin[2], Bo Sun[6], Dantong Yu[3]



- Where are the model and trained weights?
- How do I run the model on my data?
- Should I run the model on my data?
- How can I retrain the model on new data?
- How can I build on this work?
- How do I share my model with the community?





Model / transform containers

ARGONNE LEADERSHIP COMPUTING FACILITY

parsl

MATERIALS DATA FACILITY

PETREL

DLHub and FuncX

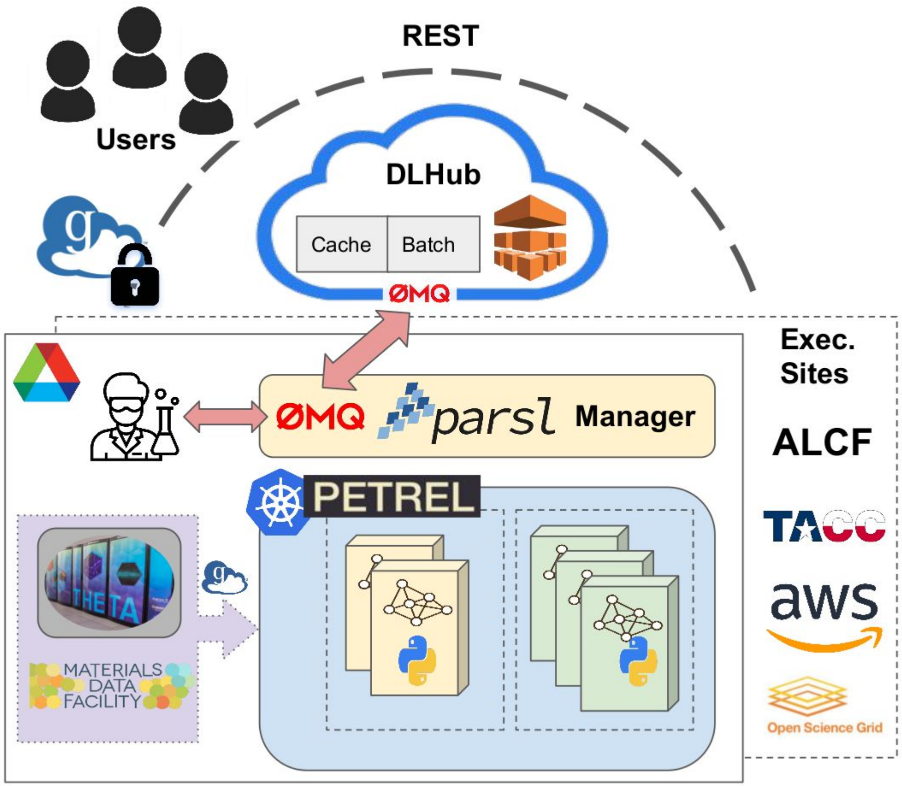
Vanguard model dependencies necessitate containerization

Real-time usage relies on low-latency

Serving works well on kubernetes

Training requires HPC resources

FuncX fulfills both inference serving and model training requirements; manages servables (runtime + shim); and enables low latency invocation



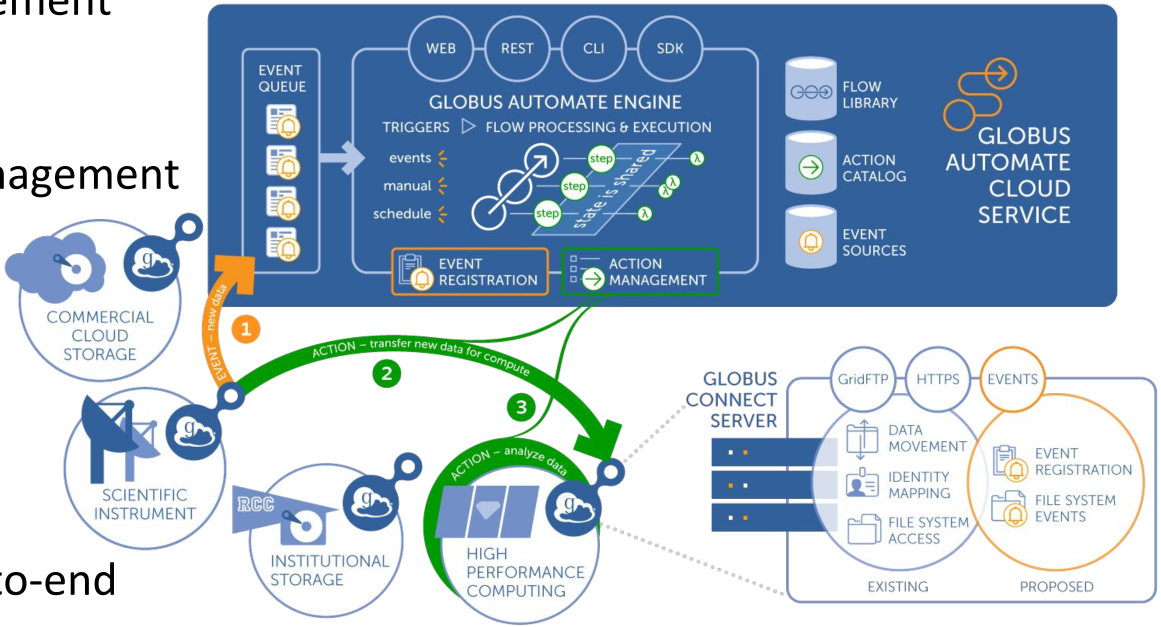
Automate

Distributed research data management automation

Construct “pipelines” of data management tasks, e.g.:

- Transfer
- Catalog
- Set ACLs
- Share

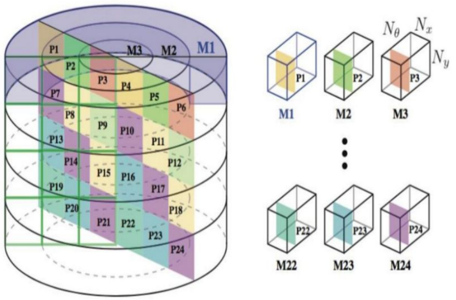
Can be used for automating end-to-end analysis pipelines

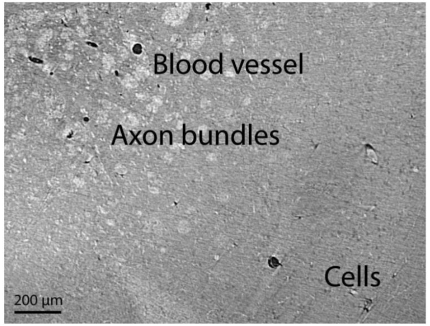


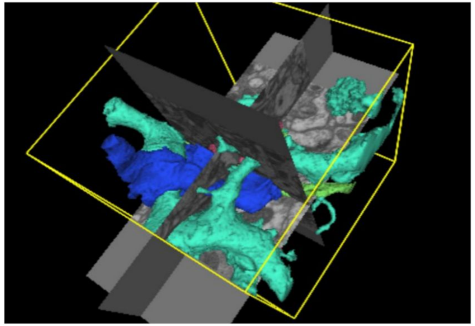
Neuroanatomy

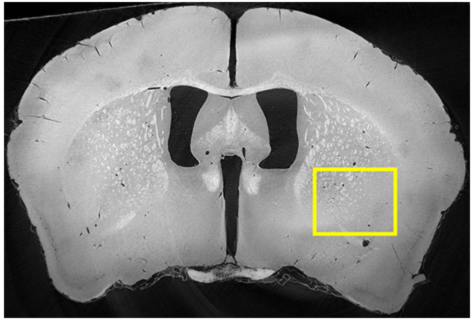
UChicago’s Kasthuri Lab study brain aging and disease

- Construct connectomes -- mapping of neuron connections
- Use synchrotron (APS) to rapidly image brains (and other things)
- Given beam time once every few months
- Generate segmented datasets/visualizations for the community
- ~20GB/minute for large (cm) unsectioned brains
- Perform semi-standard reconstruction on all data across HPC resources

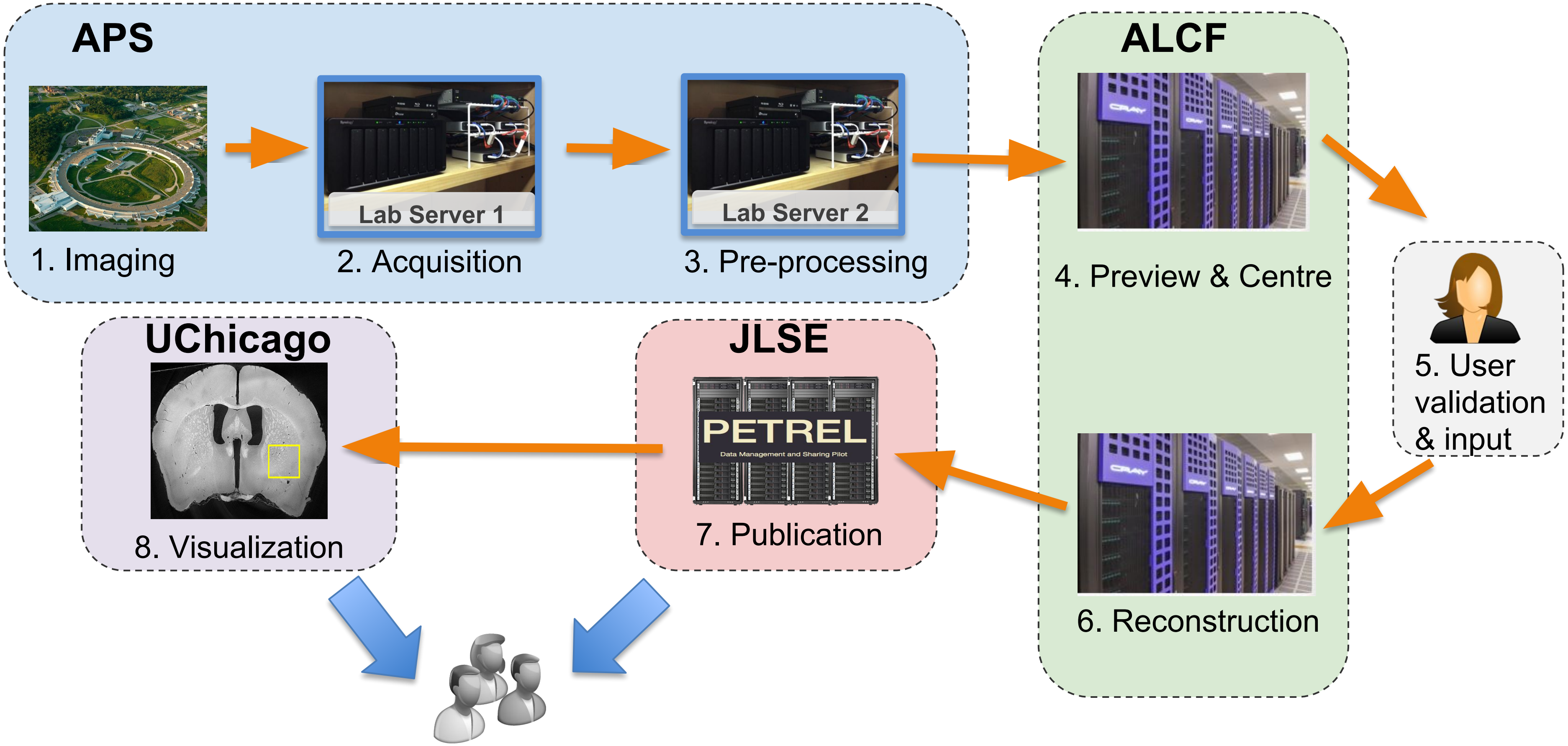




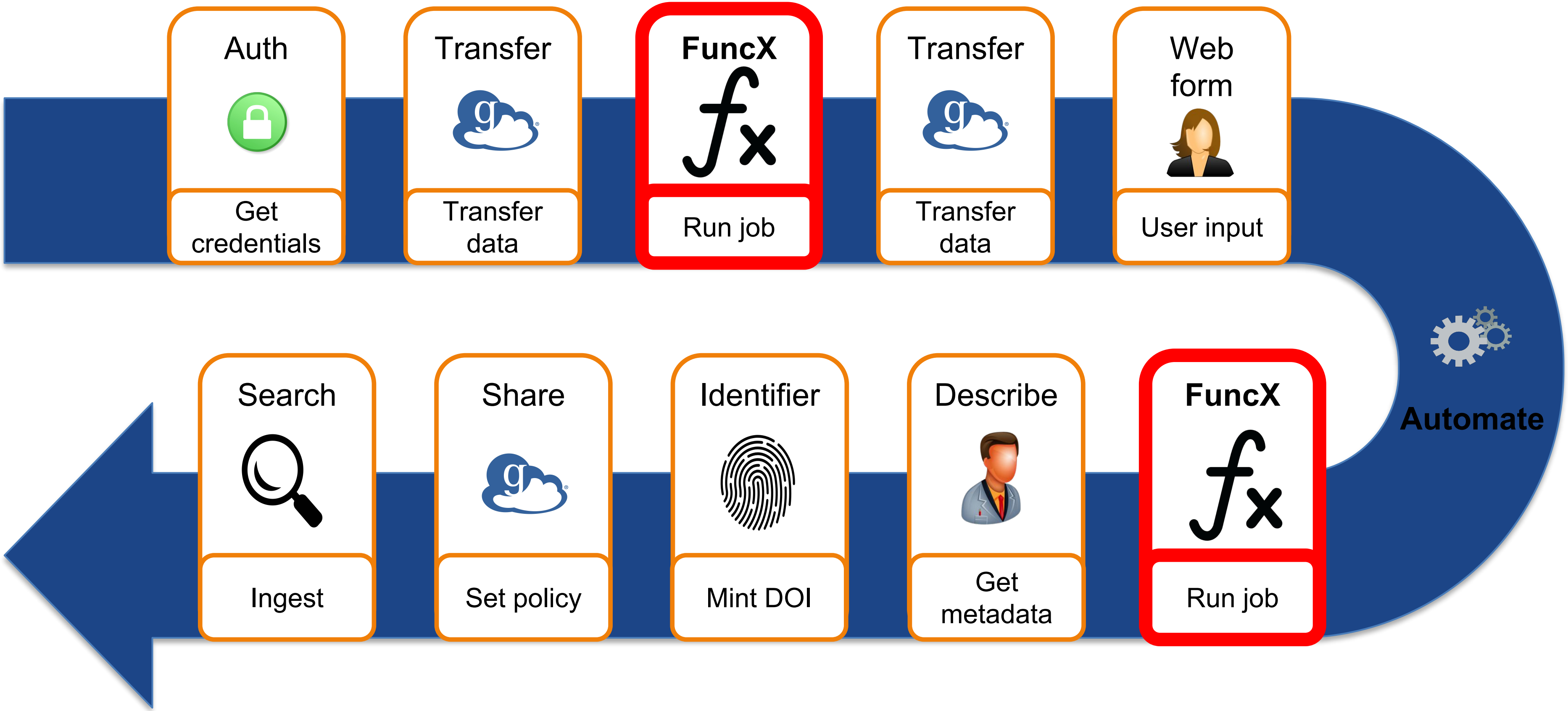




Neuroanatomy automation



Neuroanatomy automation



Automate and FuncX

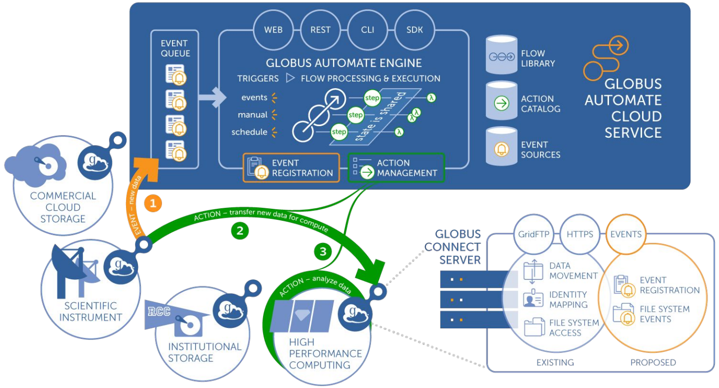
Secure and reliable remote execution platform

Containerized reconstruction functions can be reused between datasets

Enables reproducibility and sharing of pipelines between beamlines

On-demand analysis when the beam is running

Abstracts HPC integration for domain scientists



Extract

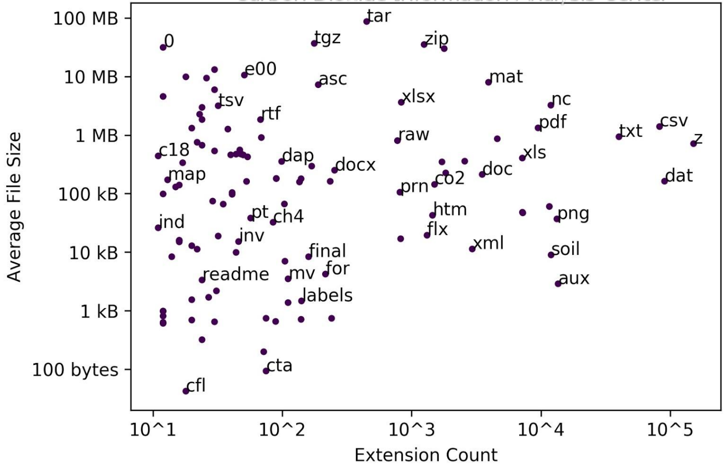
File systems and data repositories are often inconsistent and messy

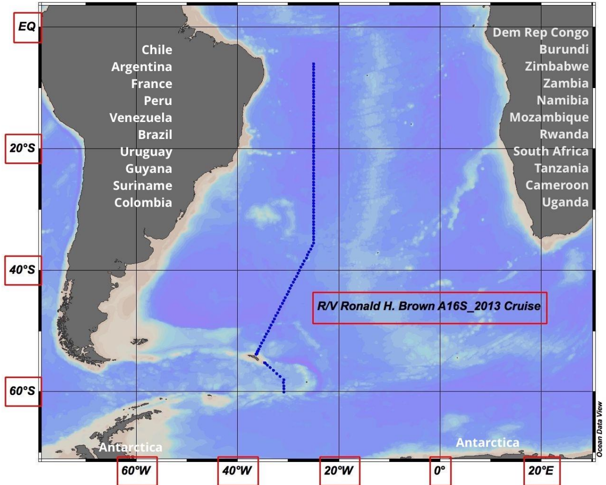
Extract aims to ***drain the data swamp***

Analyze files, extract metadata, catalog data

Dynamic pipelines to maximize searchable metadata extracted

- Modular extractors are pipeline steps
  - Apply many extractors to each file
  - Different files require different extractors
- Prioritize extractors by expected yield





Extract and FuncX

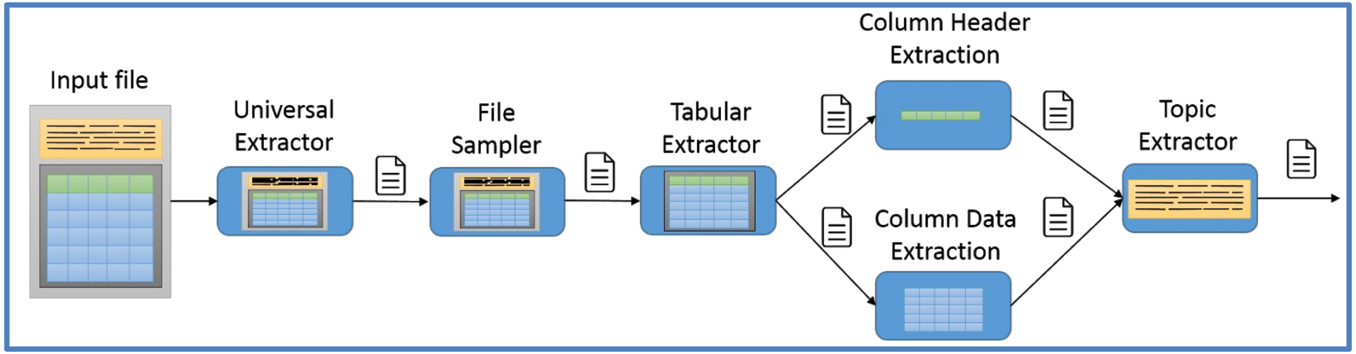
Running at scale (PB store) requires invocation at data

FuncX manages deployment and invocation of extractors

Can be run as compute is available (backfill)

Run in response to data events (files created/changed)

Push extractor functions to arbitrary machines



Future work

Complete node manager implementation

Identify new use cases (and requirements)

Investigate multi-tenant solutions

Thanks

Questions, comments, use cases?

rchard@anl.gov