funcX

funcX: A Federated Function Serving Fabric for Science

Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard,
Ben Blaiszik, Ben Galewsky, Daniel S. Katz, Ian Foster, Kyle Chard

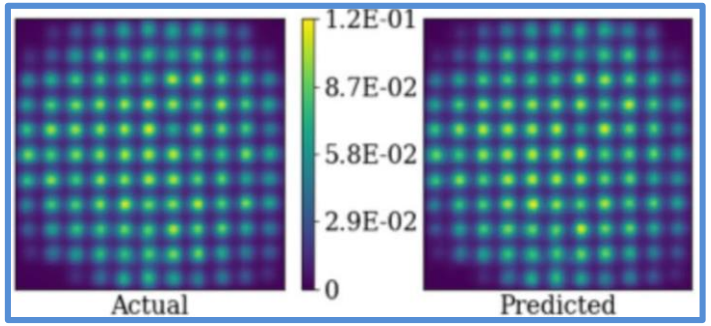
THE UNIVERSITY OF CHICAGO

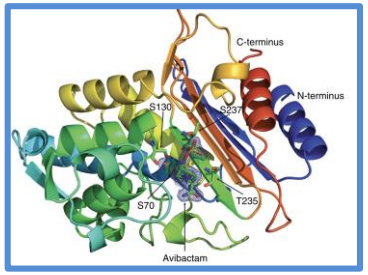
ILLINOIS

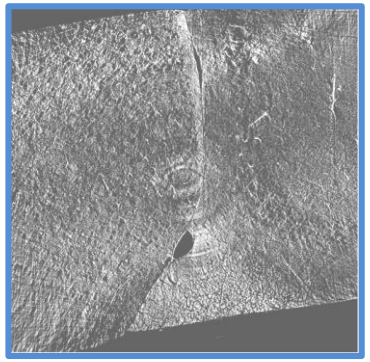
Argonne NATIONAL LABORATORY

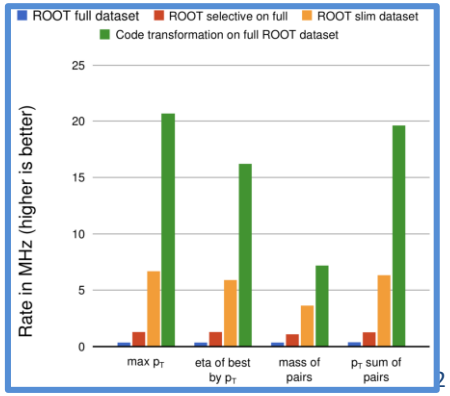
The need for remote computing

- We have long strived to compute wherever it makes the most sense:
  - Resource availability, data location, analysis time, wait time, software licenses, etc.
- Remote computing has been complex and expensive; however we now have:
  - High speed networks
  - Universal trust fabrics
  - Containers









Serverless computing

Provider runs infrastructure and manages allocation of resources

Function as a Service (FaaS)

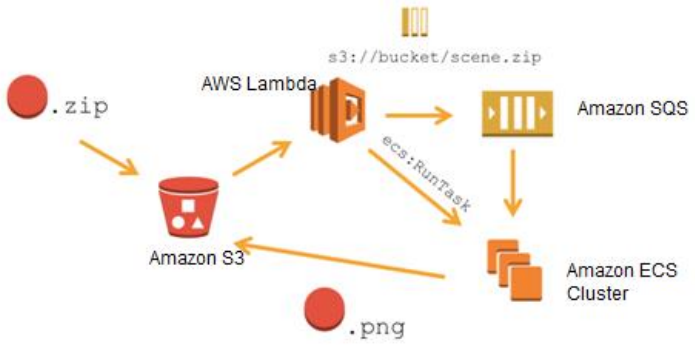
- Pick a runtime (Python/JS/R etc.)
- Write function code
- Run (and scale)

Low latency, on-demand, elastic scaling

Combine functions (e.g., workflows) to solve complex problems

Google Cloud Functions

APACHE OpenWhisk™



Adapting function as a service for science?

1. Support new workloads by decomposing applications into functions
   - Real-time, interactive, stream processing
   - Simplify development, maintenance, testing
2. Facilitate use of diverse compute resources
   - Abstract compute infrastructure
3. Enable fluid function execution across the heterogeneous computing continuum
   - Containers enable portability and sandboxing

➔ funcX: high performance and federated function as a service

FuncX: a federated function serving ecosystem for science

Functions:

- Register once, run anywhere, any time

Endpoints:

- Dynamically provision resources, deploy containers, and execute functions
- Exploit local architecture/accelerators

funcX Service:

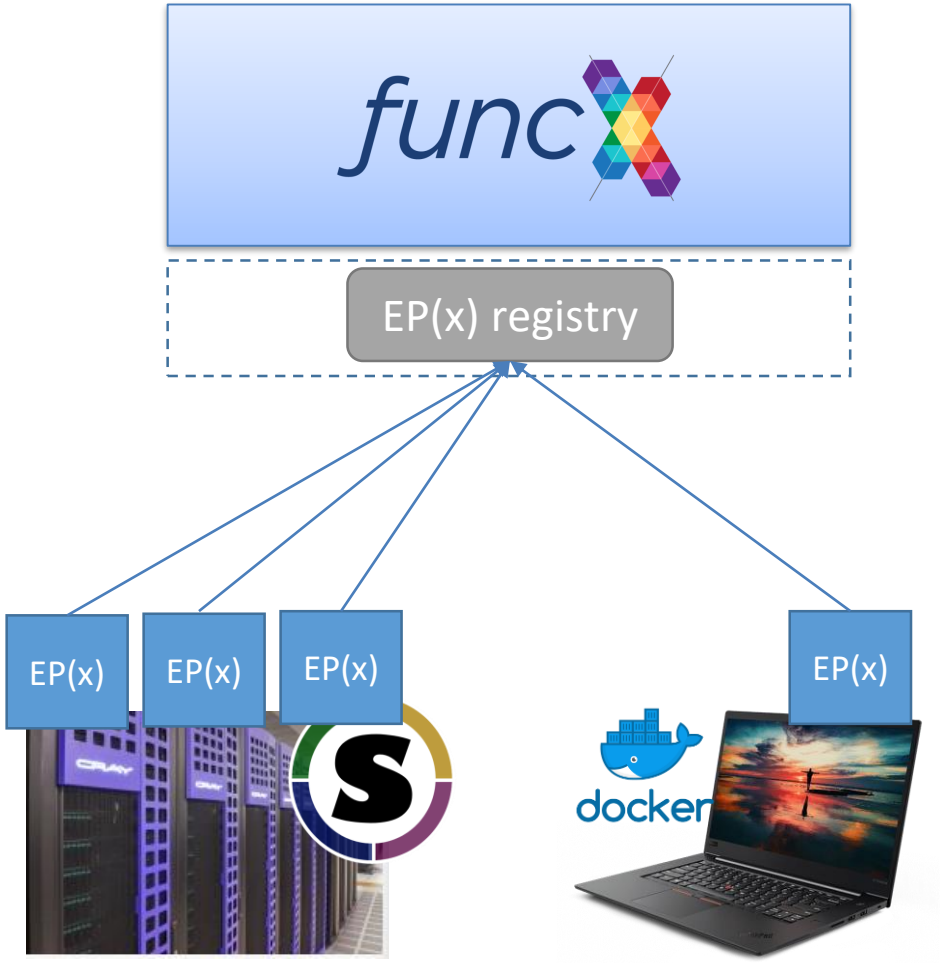
- Register and share endpoints
- Register, share, run functions

Turn ***any*** machine into a function serving endpoint

Route functions to remote endpoints

- Closest, cheapest, fastest, accelerators …

Transform clouds, clusters, and supercomputers into high-performance function serving systems



6

Register functions for execution on any funcX endpoint

Registration

f(x) + dependencies

```
def funcx_sum(items):
    return sum(items)
```

funcX

EP(x) registry

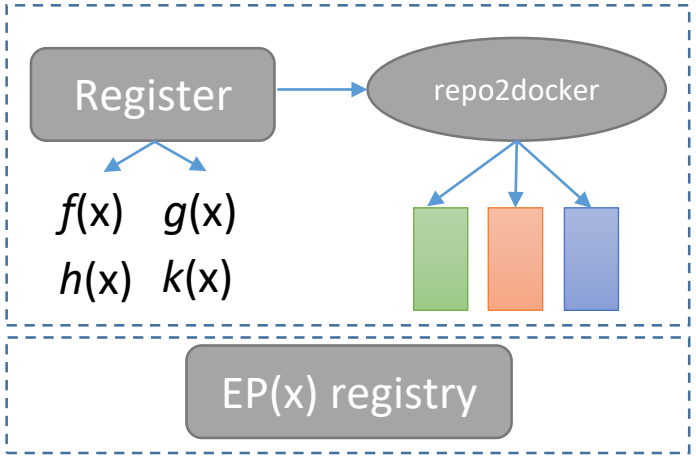
7

Register functions for execution on any funcX endpoint

Registration

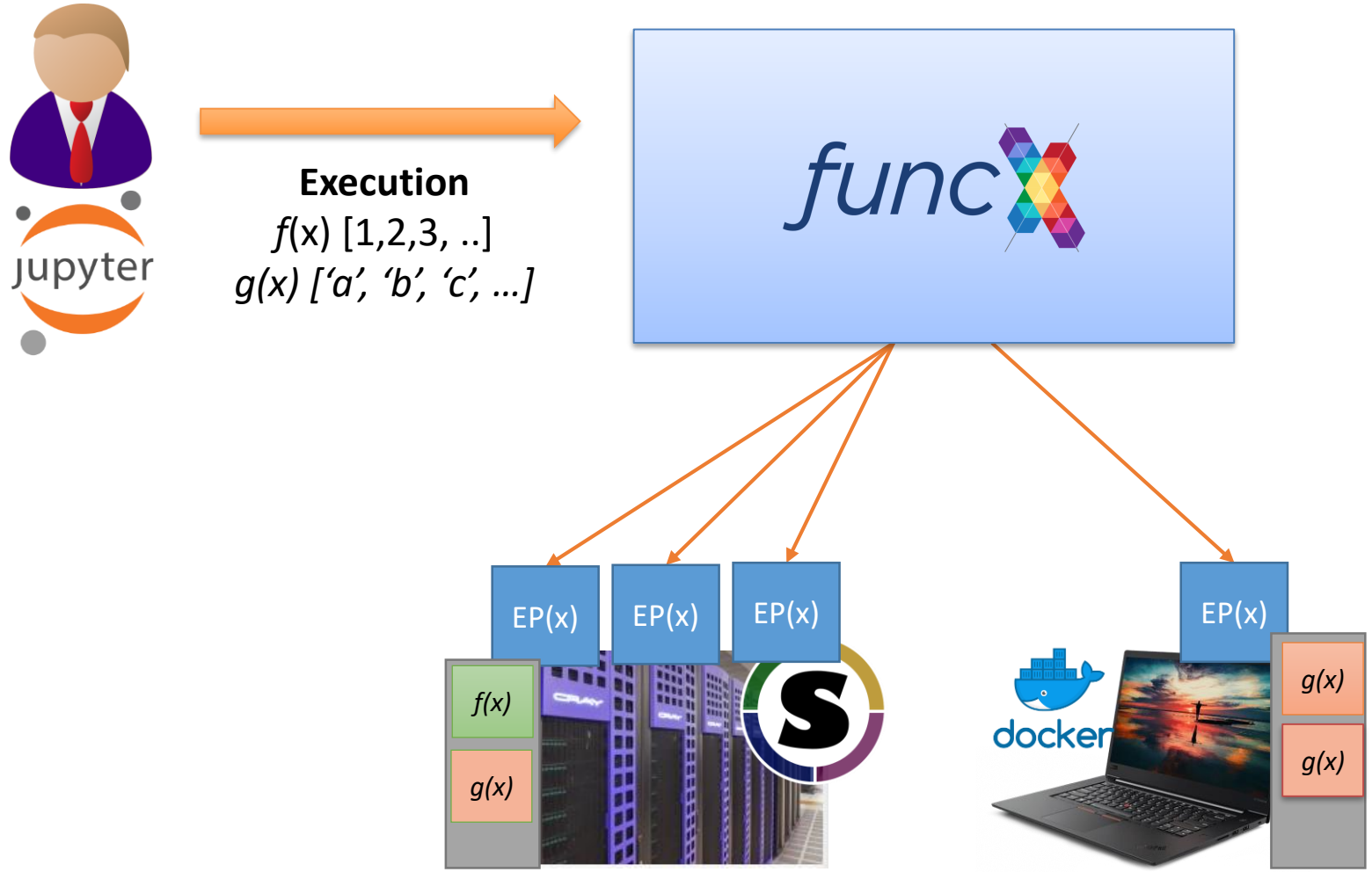
f(x) + dependencies

```
def funcx_sum(items):
    return sum(items)
```



8

Reliably and scalably execute registered functions on any funcX endpoint



9

Deploying a funcX endpoint

- Pip install funcX (e.g., using Conda) in user space
- Authenticate and register with the funcX service
- Configure the endpoint for the local resources (using Parsl)

```
from funcx.config import Config
from parsl.providers import SlurmProvider
from parsl.launchers import SrunLauncher

config = Config(
    provider=SlurmProvider(
        'debug',
         launcher=SrunLauncher(),
         nodes_per_block=5,
         init_blocks=1,
         min_blocks=1,
         max_blocks=1,
         worker_init='source activate funcx',
         walltime='00:30:00',
    ),
    max_workers_per_node=28,
)
```

XSEDE

Google Cloud

Open Science Grid

amazon web services

Demo

Setup an endpoint

$ pip install funcx

$ funcx-endpoint configure <ENDPOINT_NAME>

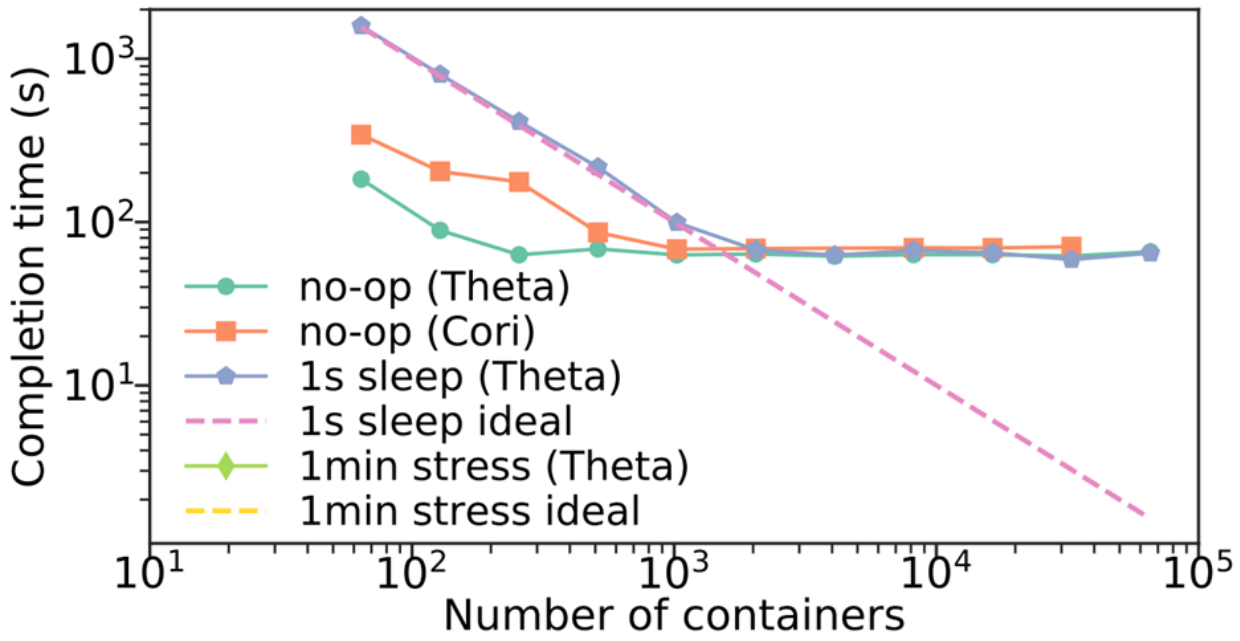
$ funcx-endpoint start <ENDPOINT_NAME>

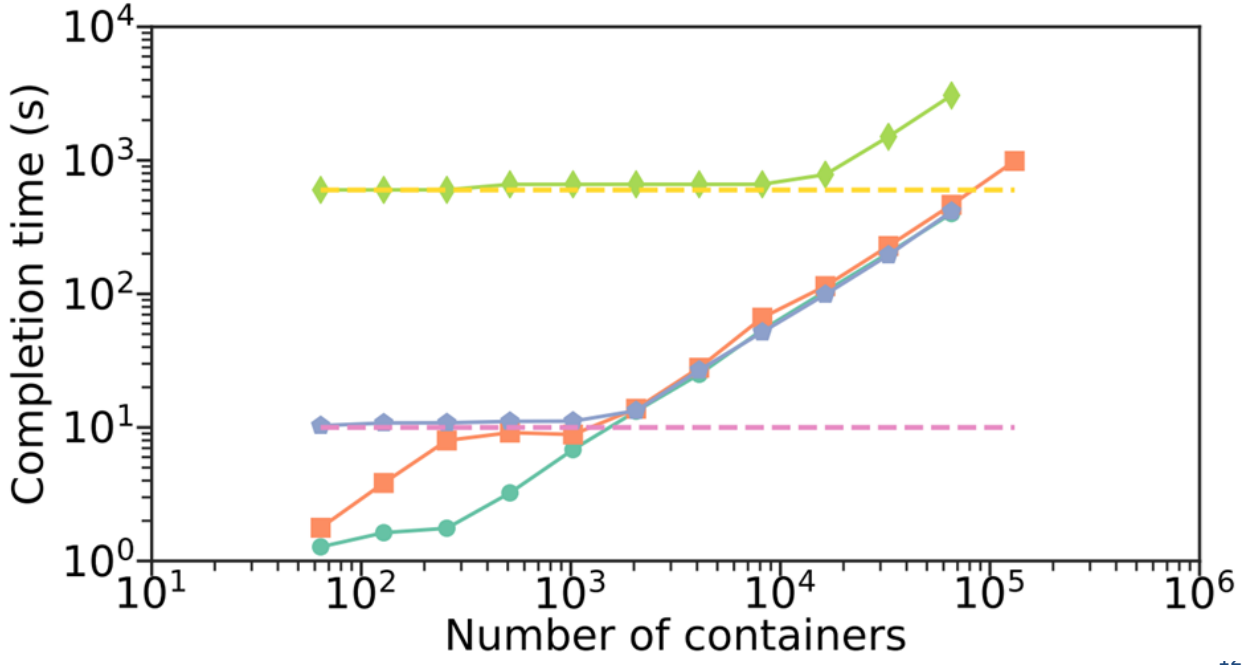
Run a function

```
from funcx.sdk.client import FuncXClient
fxc = FuncXClient()

def funcx_sum(items):
    return sum(items)

func_uuid = fxc.register_function(funcx_sum)

res = fxc.run(items, endpoint_id=<UUID>,
function_id=func_uuid)

fxc.get_result(res)
```

funcX scales to 100K+ workers

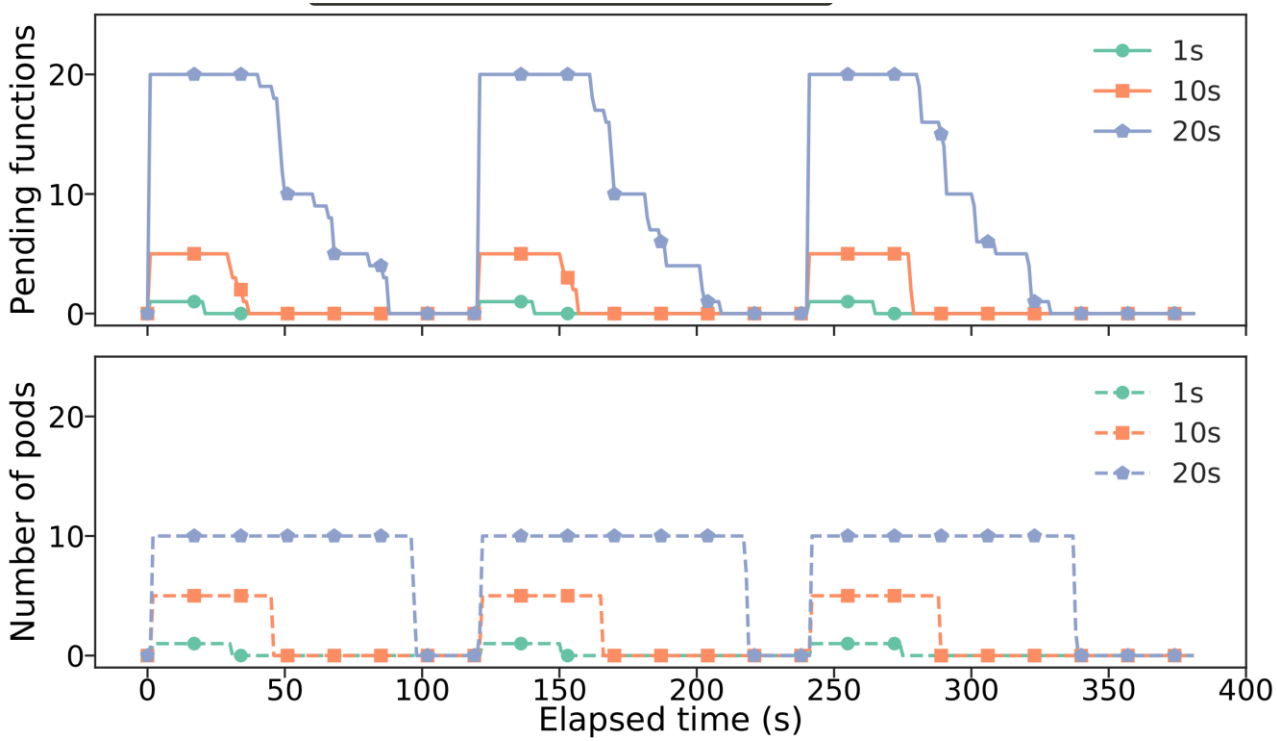
- funcX endpoints deployed on ALCF Theta and NERSC Cori
- Strong scaling (100K concurrent functions) shows good scaling up to 2K containers even with short no-op/sleep tasks
- Weak scaling (10 tasks per container) shows scaling to 131K concurrent containers (1.3M tasks)



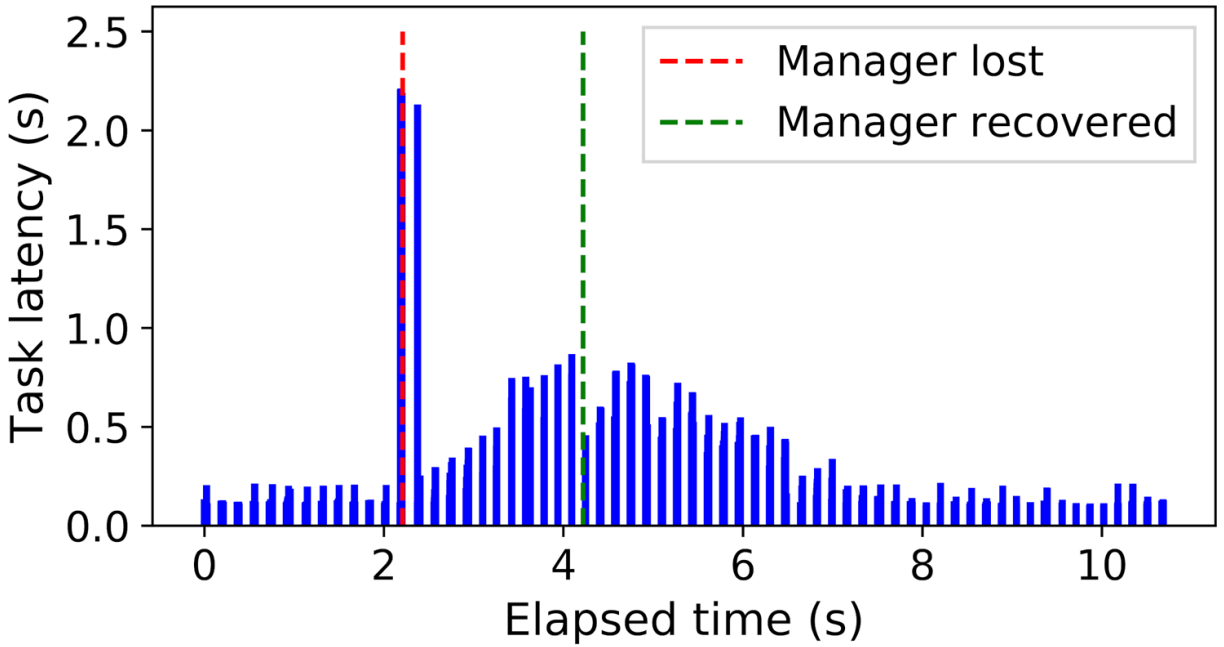


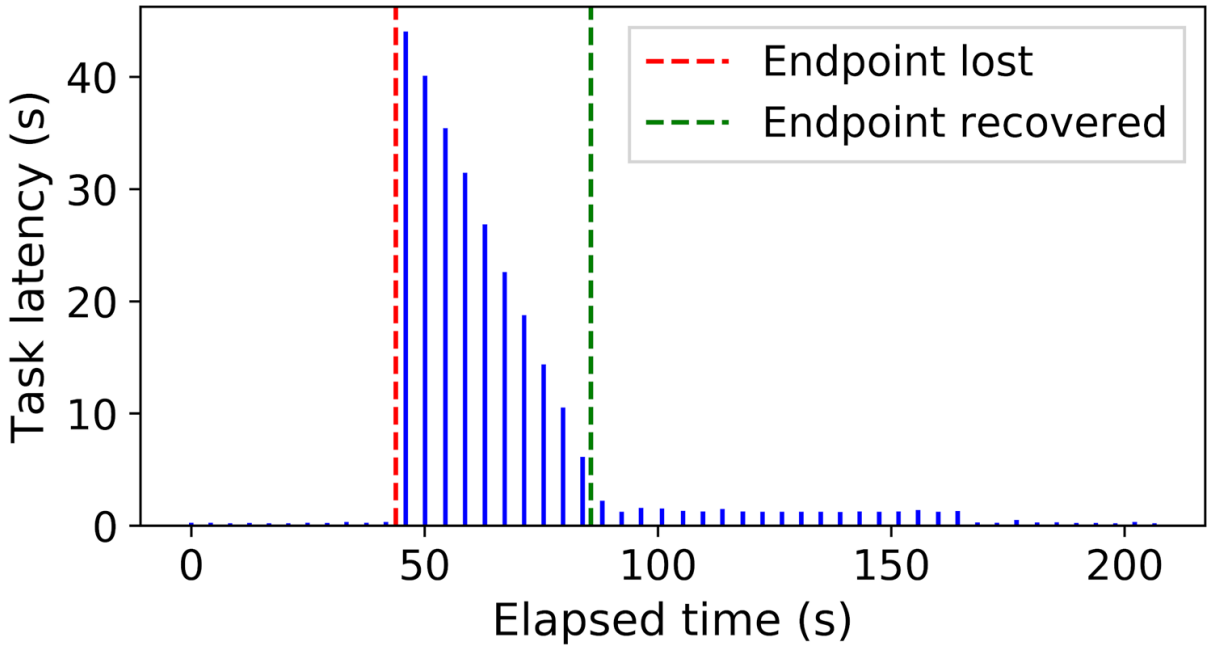
Elastic execution irrespective of underlying system

- funcX agent deployed on a Kubernetes cluster
- Each function is registered in a container and allowed to use 0-10 pods (unit of execution)
- FuncX elastically scales active pods (bottom) based on workload (top)



funcX recovers from worker, manager, and endpoint failures

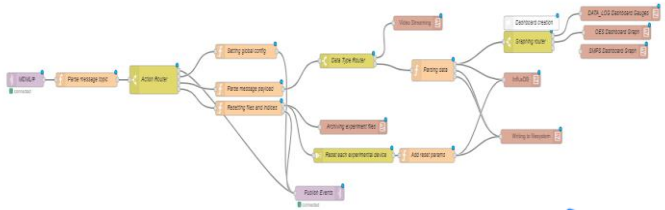




Example application: Manufacturing

Flame spray pyrolysis, MERF

The Manufacturing and ML platform (**MDML**)



MINIO

influxdb

Grafana Real-Time Dashboards

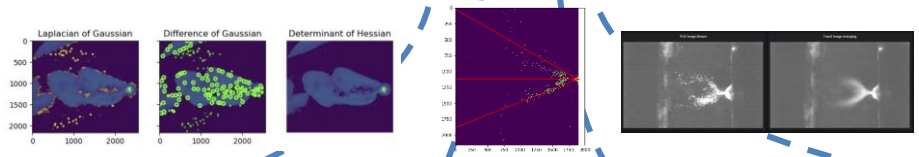
1. Instrument sensors stream data to the MDML

2. Use FaaS to analyze data on-demand

f(X) funcX

3. FaaS tasks distributed across the computing continuum

4. Results are used to guide the experiment



Edge devices

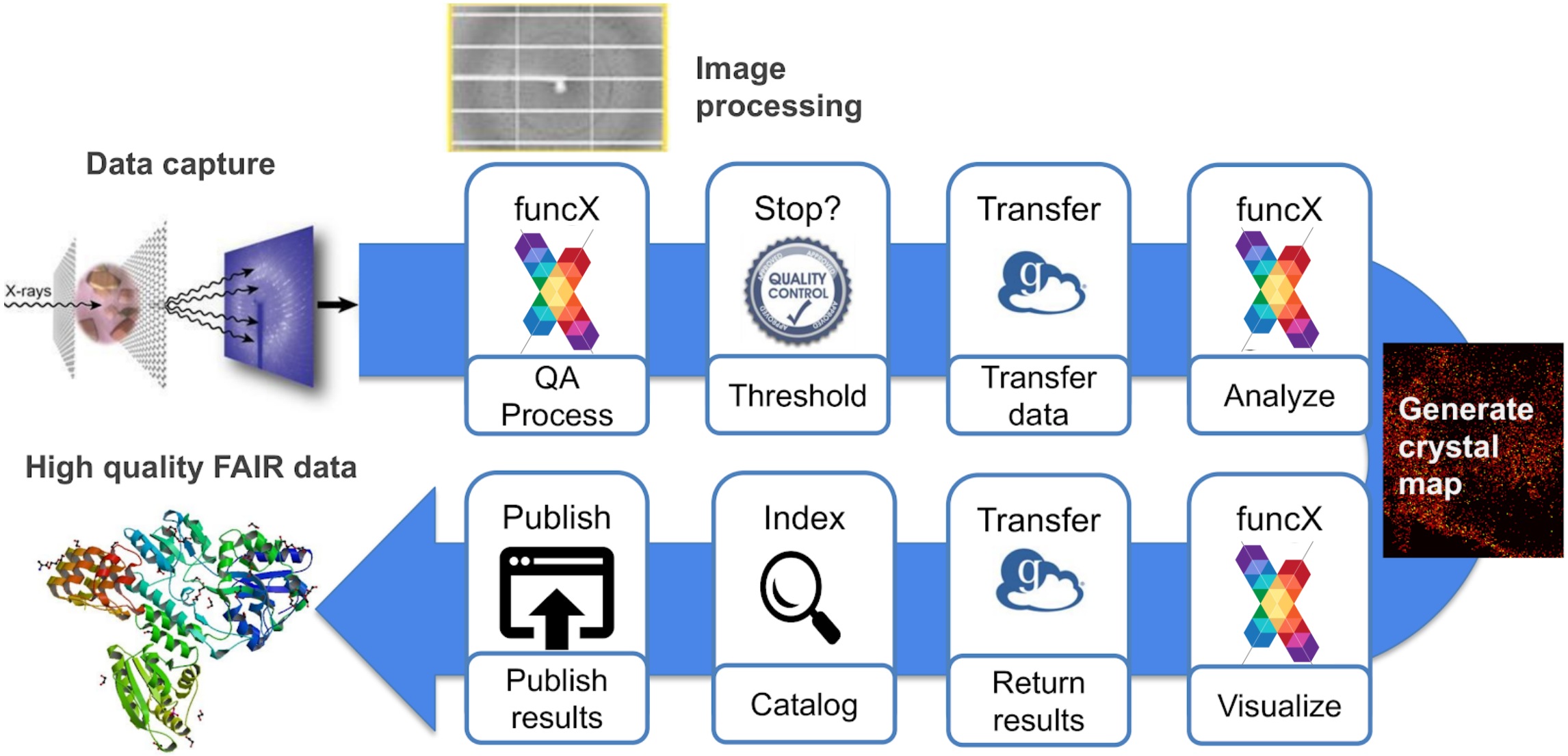
Laboratory machines

HPC

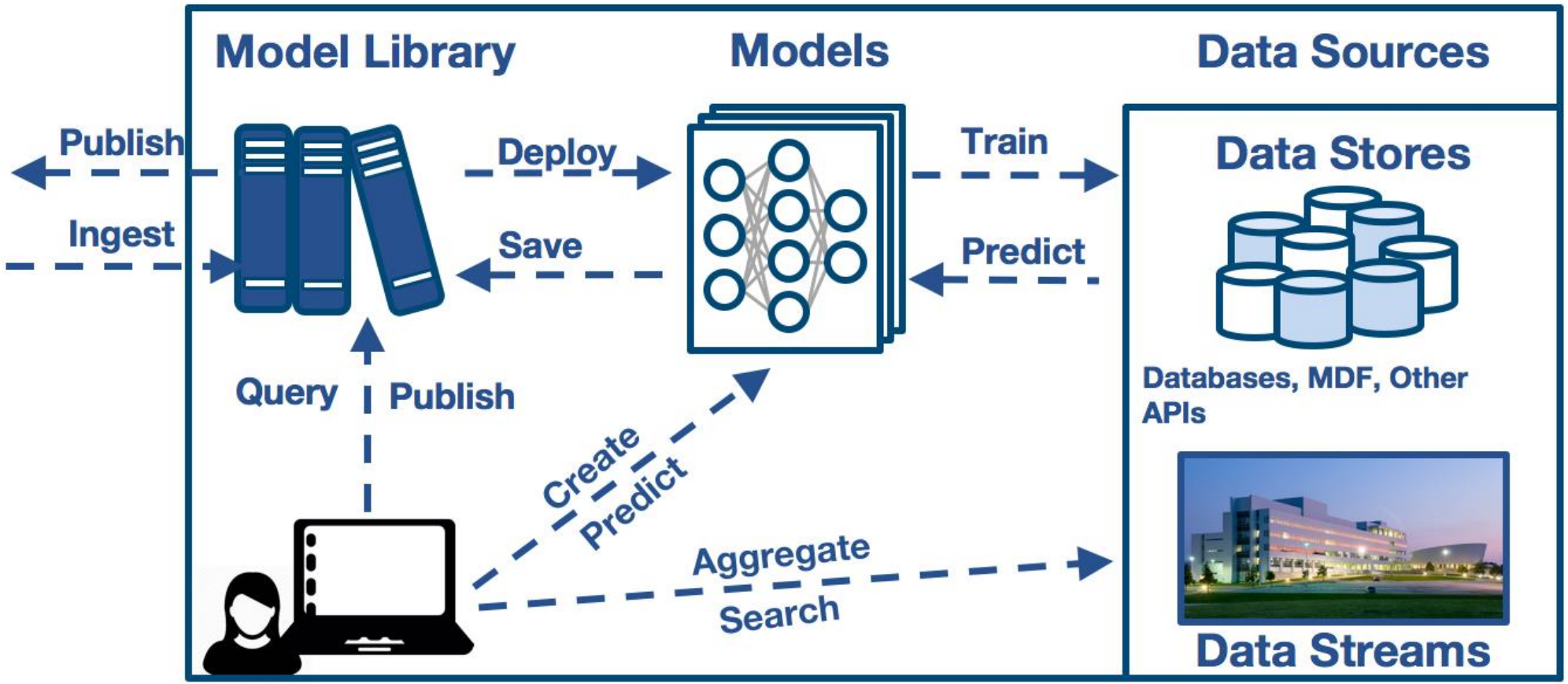
aws

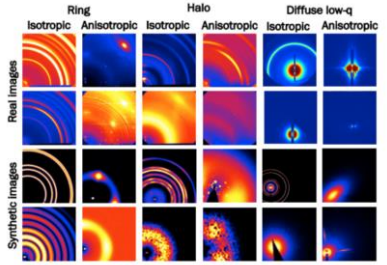
Compute and storage continuum

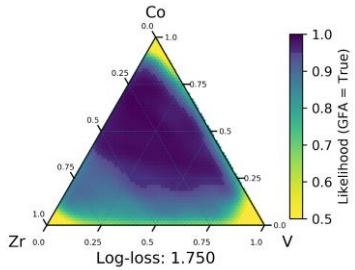
Example application: Serial Crystallography

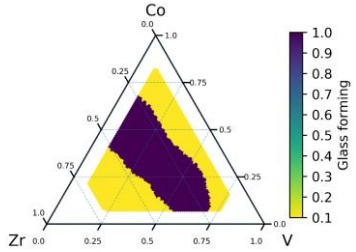


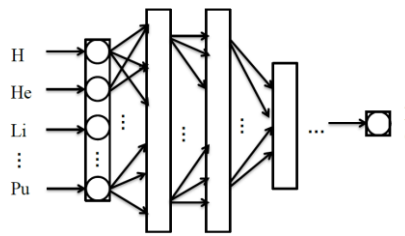
Example application: DLHub









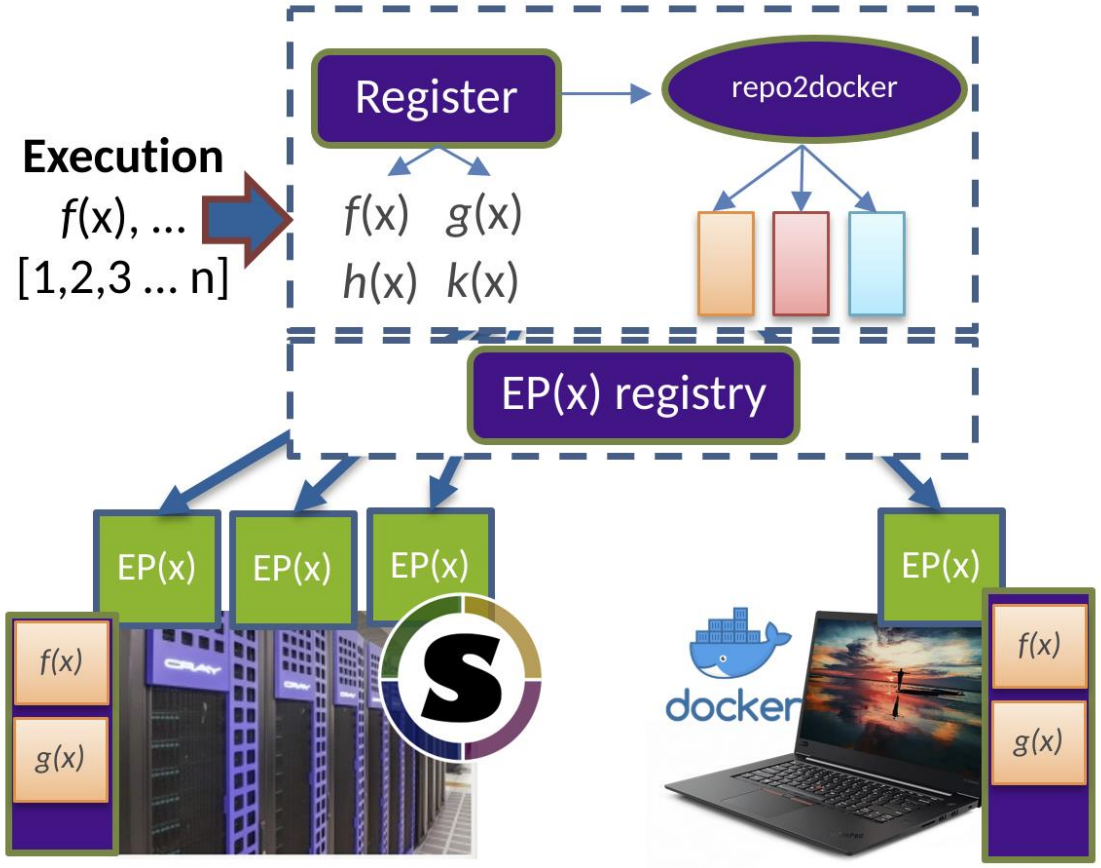


funcX creates a federated FaaS ecosystem for science

funcX is a federated FaaS system designed to meet the requirements of scientific computing

Enables fluid execution by dispatching functions to wherever makes the most sense

Initial deployments scale to 130K+ concurrent workers and >1.2M functions



http://github.com/funcx-faas

funcX

http://funcx.org

https://mybinder.org/v2/gh/funcx-faas/funcx/master