# Portable serverless computing to enable scalable data science

*Yadu Babuji[*], Josh Bryan[*⚑], Kyle Chard[*⚑], Ryan Chard[*⚑], Ben Clifford[*], Ian Foster[*⚑], Ben Galewsky[°], Daniel S. Katz[°], Kevin Hunter Kesling[*], Zhuozhao Li[*], Kirill Nagairtsev[*], Stephen Rosen[*], Tyler Skluzacek[*]*

*[*]University of Chicago & Argonne National Laboratory; [⚑]Globus;*

*[°]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign*

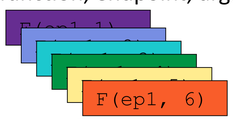
## Federated Function as a Service

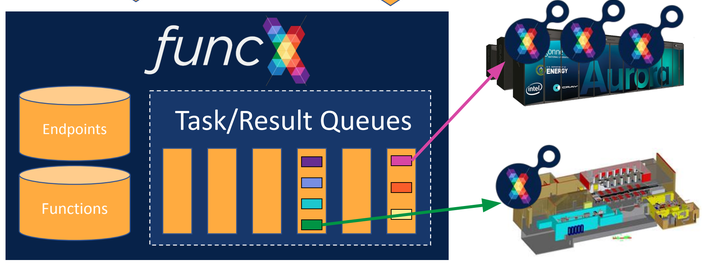
- Modern computing environments are distributed and heterogeneous; modern workloads require specialized hardware, rapid responses, and remote processing (e.g., near data)
- FaaS provides an intuitive interface for users to register and invoke programming functions without regard for underlying infrastructure
- Federated FaaS enables functions to be dispatched to remote endpoints chosen for data locality, security, or other concerns
- funcX allows users to execute Python functions (which may invoke executables, MPI programs, etc.) on arbitrary resources (e.g., CPUs, GPUs) from short to long run times

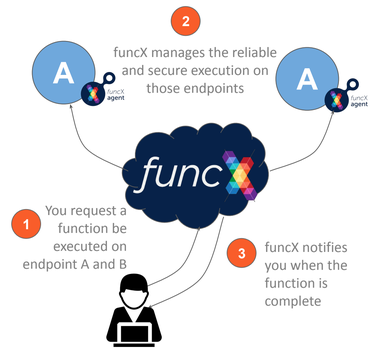
**(1) Registration**
(function + container)

```
def compute(args):
  # do something
  return results
```

**(2) Execution**
(function, endpoint, args)




## Fire-and-forget execution

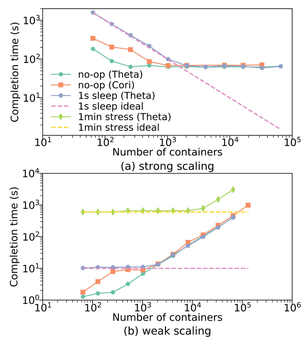
### Outsource the challenging aspects of remote execution

funcX manages authentication, serialization of functions and data, reliable execution optionally in containers, and delivery of results back to requesting users

## Transform resources into FaaS endpoints

### Easily manage execution across distributed resources

The funcX endpoint software can be deployed on laptops, clouds, clusters, and supercomputers. It provisions resources elastically based on workload.

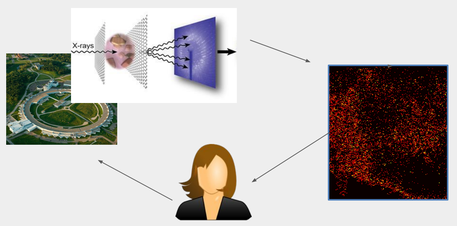

## High performance

### Launch millions of tasks

funcX supports batch submission and monitoring, asynchronous callbacks, container warming, automated resource scaling, fault tolerance, prefetching, and memoization

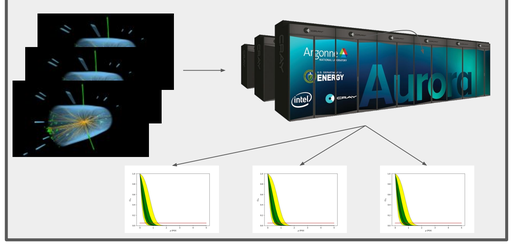
## Application examples

### Real-time crystallography




### ML-based drug screening

### HEP fitting as a service




## Install and configure a funcX endpoint

```
$ pip install funcx_endpoint

$ funcx-endpoint configure

$ funcx-endpoint start <ENDPOINT_NAME>
```

## Instantiate a funcX client

```
from funcx.sdk.client import FuncXClient

fxc = FuncXClient()
```

## Register Python function with input args

```
def hello_world():
    return "Hello World!"

func_uuid = fxc.register_function(hello_world)
```

## Execute a function by specifying endpoint and input arguments

```
ep_id = '4b116d3c-1703-4f8f-9f6f-39921e5864df'
result = fxc.run(endpoint_id=ep_id,
                 function_id=func_uuid)
```

## Retrieve results (and exceptions) asynchronously

```
fxc.get_result(result)
```

Try on Binder
https://funcx.org/binder

funcX development has been supported by NSF 2004894/2004932 and an Argonne Lab Directed Research and Development award